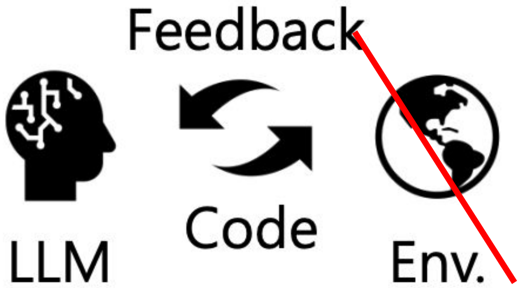
ICLR
International Conference On
Learning Representations

ConvCodeWorld: Benchmarking Conversational Code Generation in Reproducible Feedback Environments

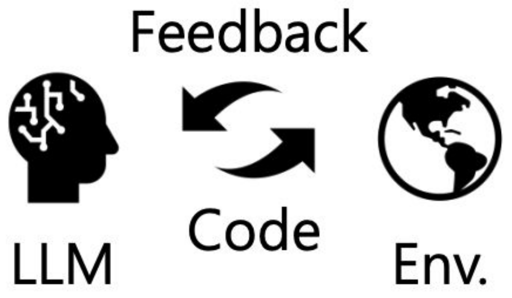
Hojae Han[1], Seung-won Hwang[1†], Rajhans Samdani[2], Yuxiong He[2]
[1]Seoul National University, [2]Snowflake AI Research
[†]Corresponding Author

Motivation: Evaluate Code LLMs on Interactable Environments

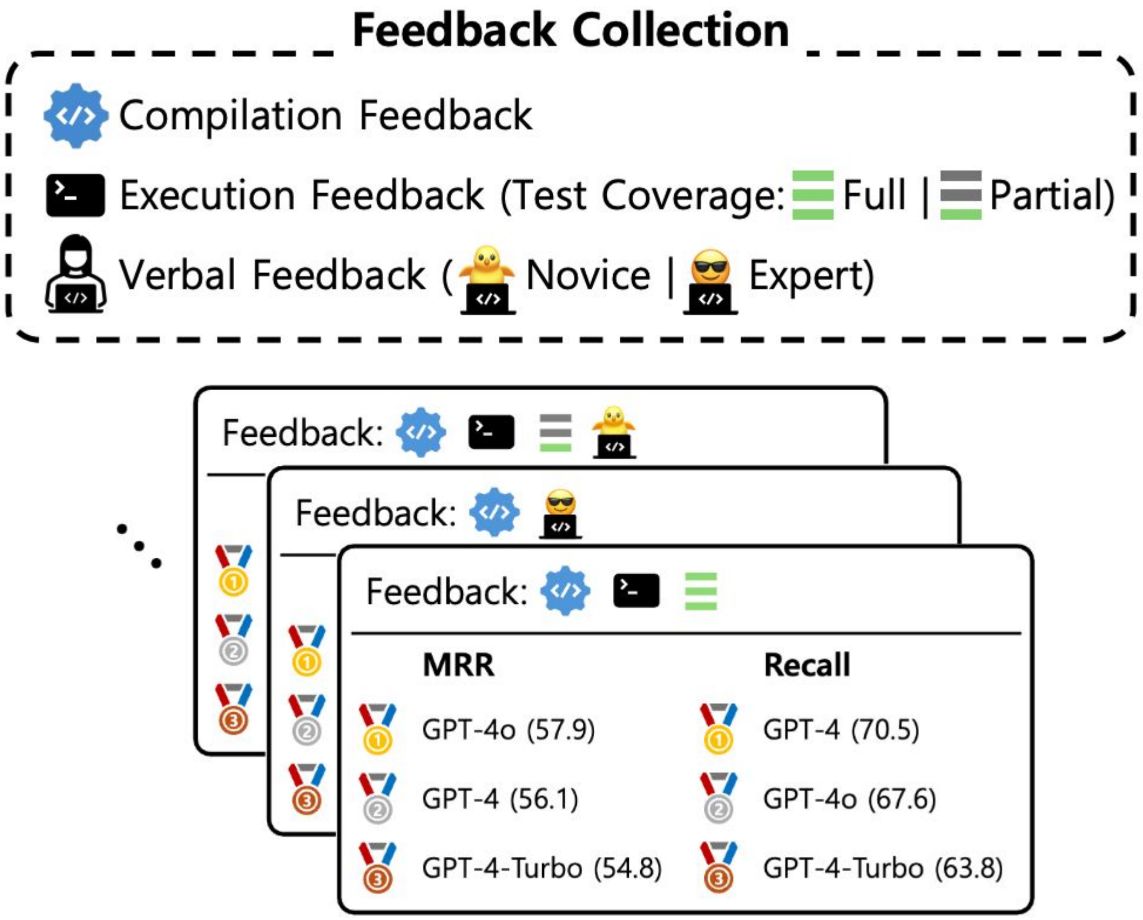


Most code generation benchmarks
do not support interactable environments.

Motivation: Evaluate Code LLMs on Interactable Environments



Our goal is to build a benchmark that can evaluate LLMs' code generation capabilities while interacting with various feedback combination.



Feedback Collection

- Compilation Feedback
- Execution Feedback (Test Coverage: ☰ Full | ☰ Partial)
- Verbal Feedback (Novice | Expert)

Feedback:

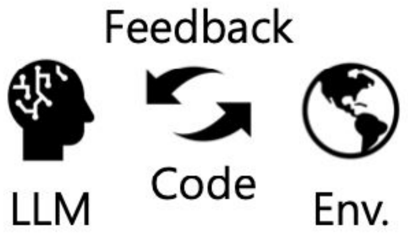
Feedback:

Feedback:

MRR	Recall
① GPT-4o (57.9)	① GPT-4 (70.5)
② GPT-4 (56.1)	② GPT-4o (67.6)
③ GPT-4-Turbo (54.8)	③ GPT-4-Turbo (63.8)

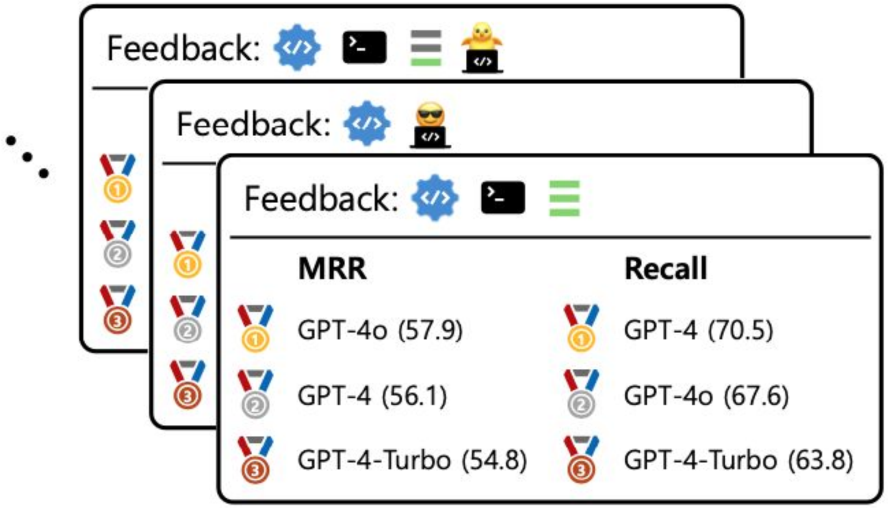
ConvCode
WORLD

A Reproducible Environment with Diverse Feedback Combinations



Feedback Collection

- Compilation Feedback
- Execution Feedback (Test Coverage: Full | Partial)
- Verbal Feedback (Novice | Expert)

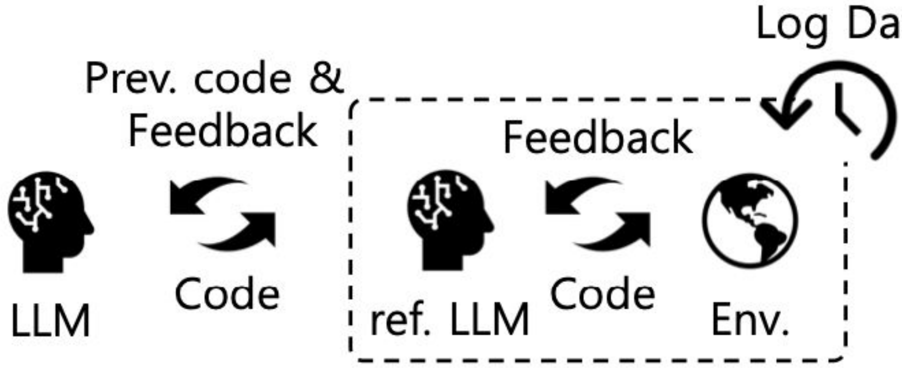


Feedback:

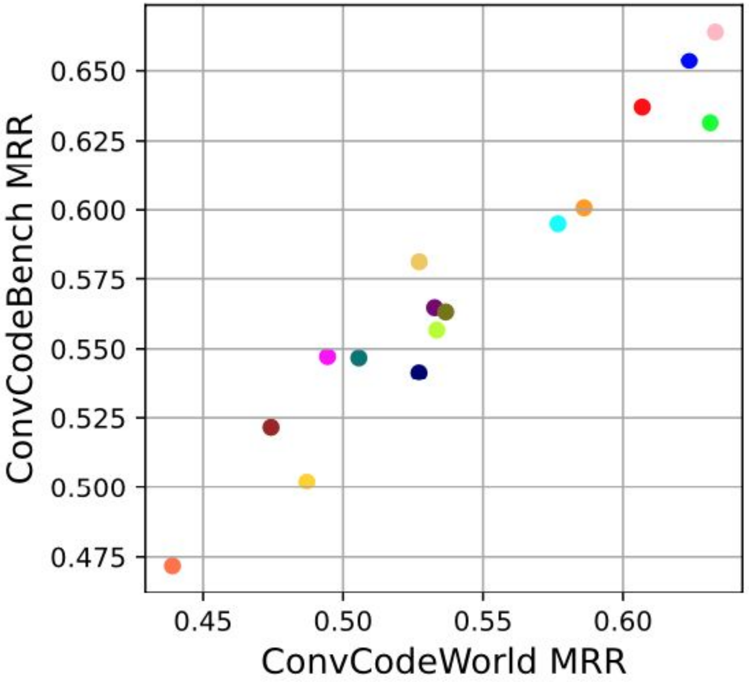
MRR	Recall
① GPT-4o (57.9)	① GPT-4 (70.5)
② GPT-4 (56.1)	② GPT-4o (67.6)
③ GPT-4-Turbo (54.8)	③ GPT-4-Turbo (63.8)

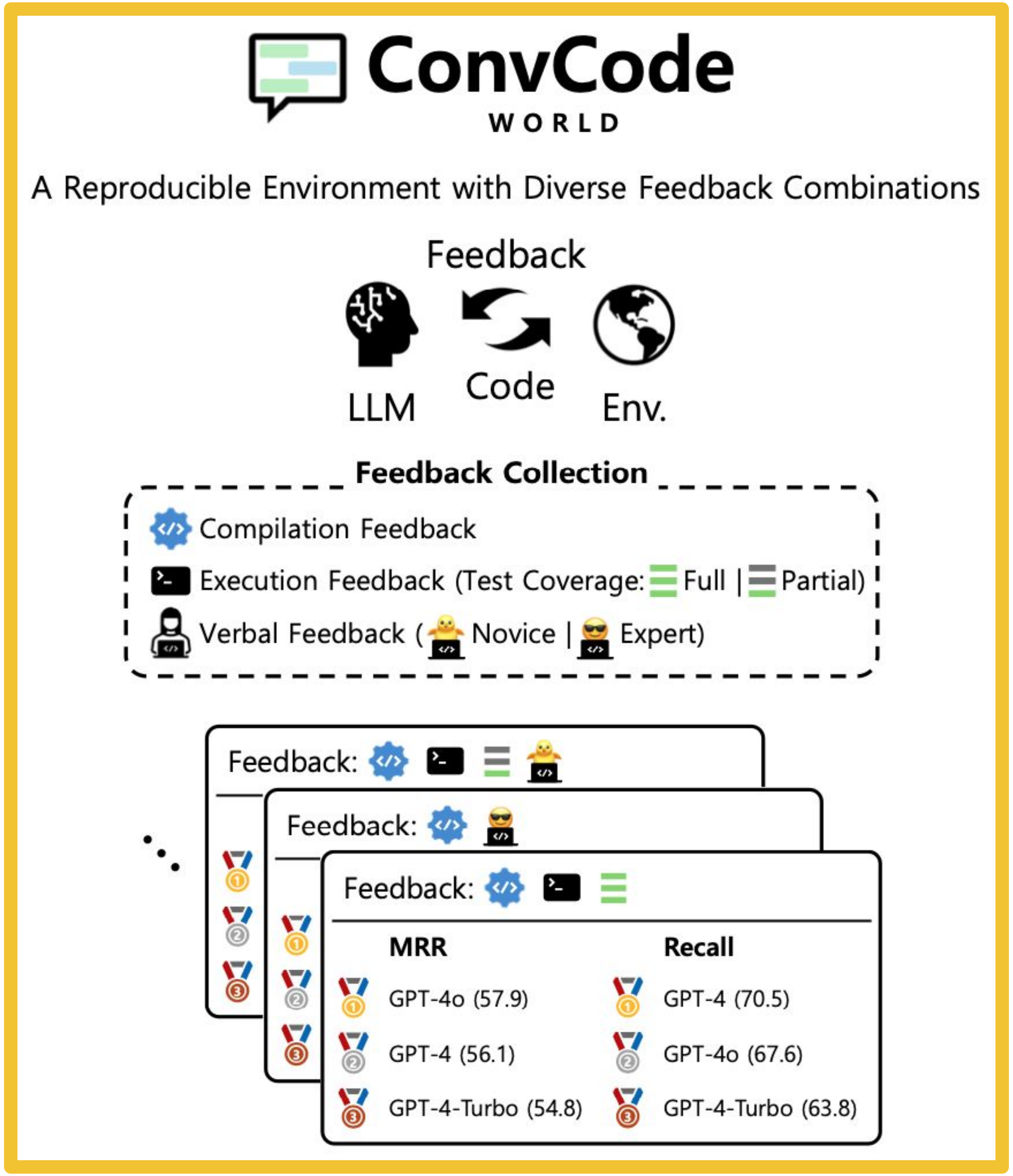
ConvCode
BENCH

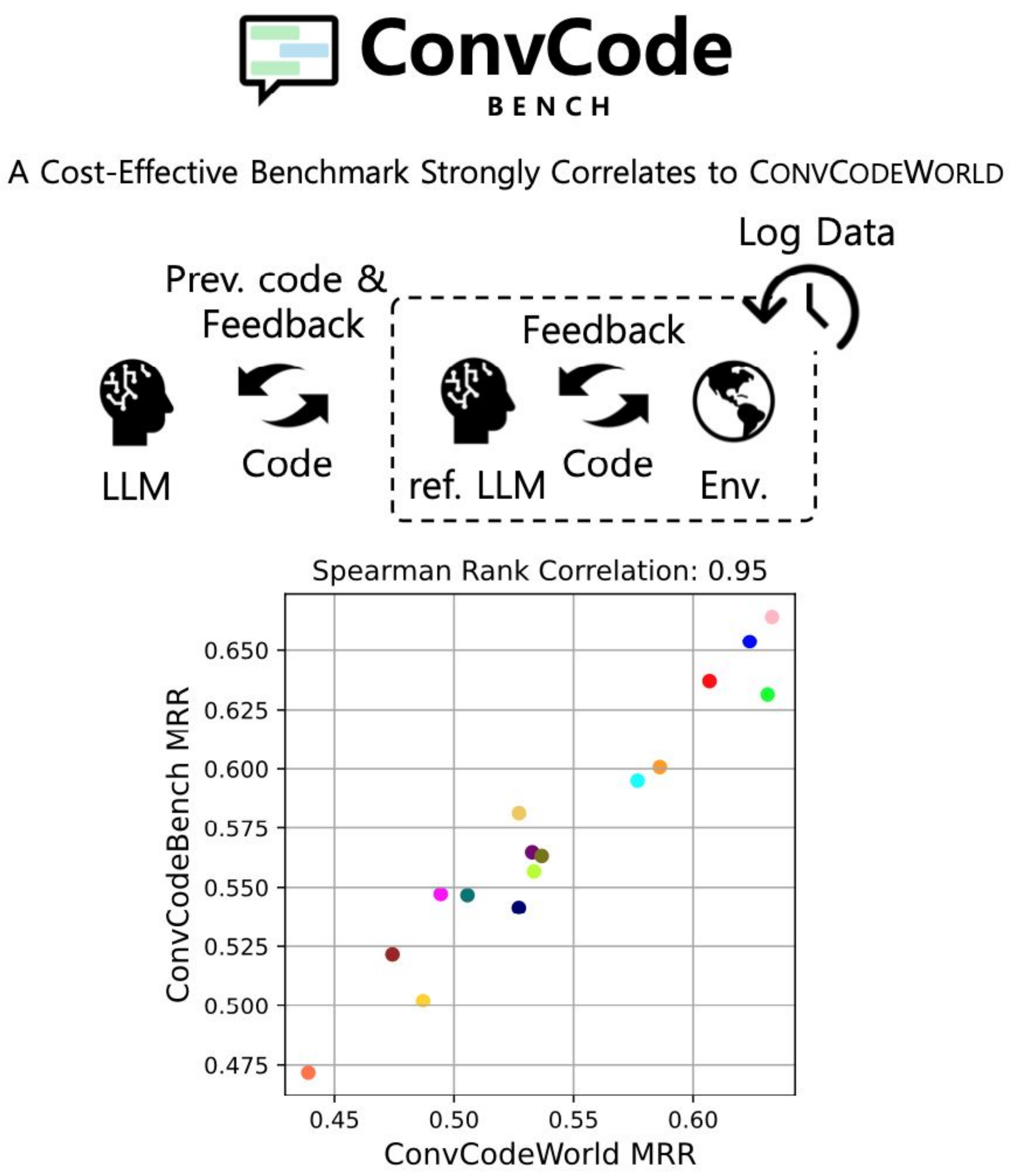
A Cost-Effective Benchmark Strongly Correlates to CONVCODEWORLD



Spearman Rank Correlation: 0.95




ConvCode
WORLD
A Reproducible Environment with Diverse Feedback Combinations
Feedback
LLM
Code
Env.
Feedback Collection
Compilation Feedback
Execution Feedback (Test Coverage: Full | Partial)
Verbal Feedback (Novice | Expert)
Feedback:
Feedback:
Feedback:
MRR
Recall
GPT-4o (57.9)
GPT-4 (56.1)
GPT-4-Turbo (54.8)
GPT-4 (70.5)
GPT-4o (67.6)
GPT-4-Turbo (63.8)

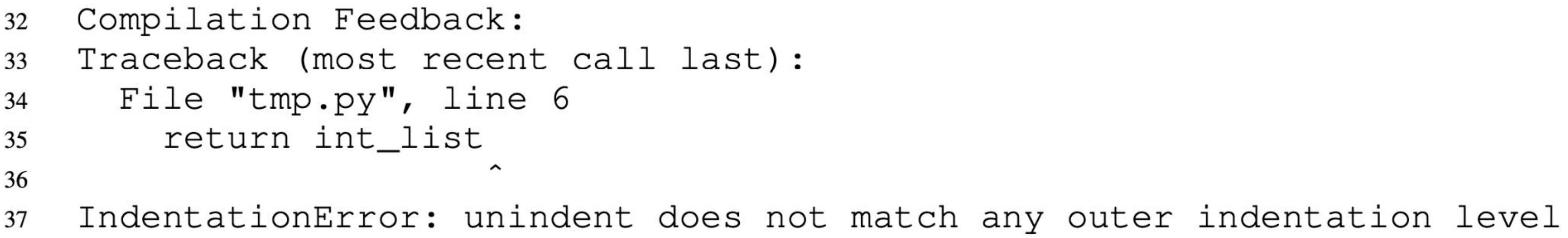

ConvCode
BENCH
A Cost-Effective Benchmark Strongly Correlates to ConvCodeWorld
Log Data
Prev. code &
Feedback
Feedback
LLM
Code
ref. LLM
Code
Env.
Spearman Rank Correlation: 0.95
ConvCodeBench MRR
0.650
0.625
0.600
0.575
0.550
0.525
0.500
0.475
0.45
0.50
0.55
0.60
ConvCodeWorld MRR

Types of Feedback

- Compilation Feedback (f_c)

Types of Feedback

- Compilation Feedback (f_c)



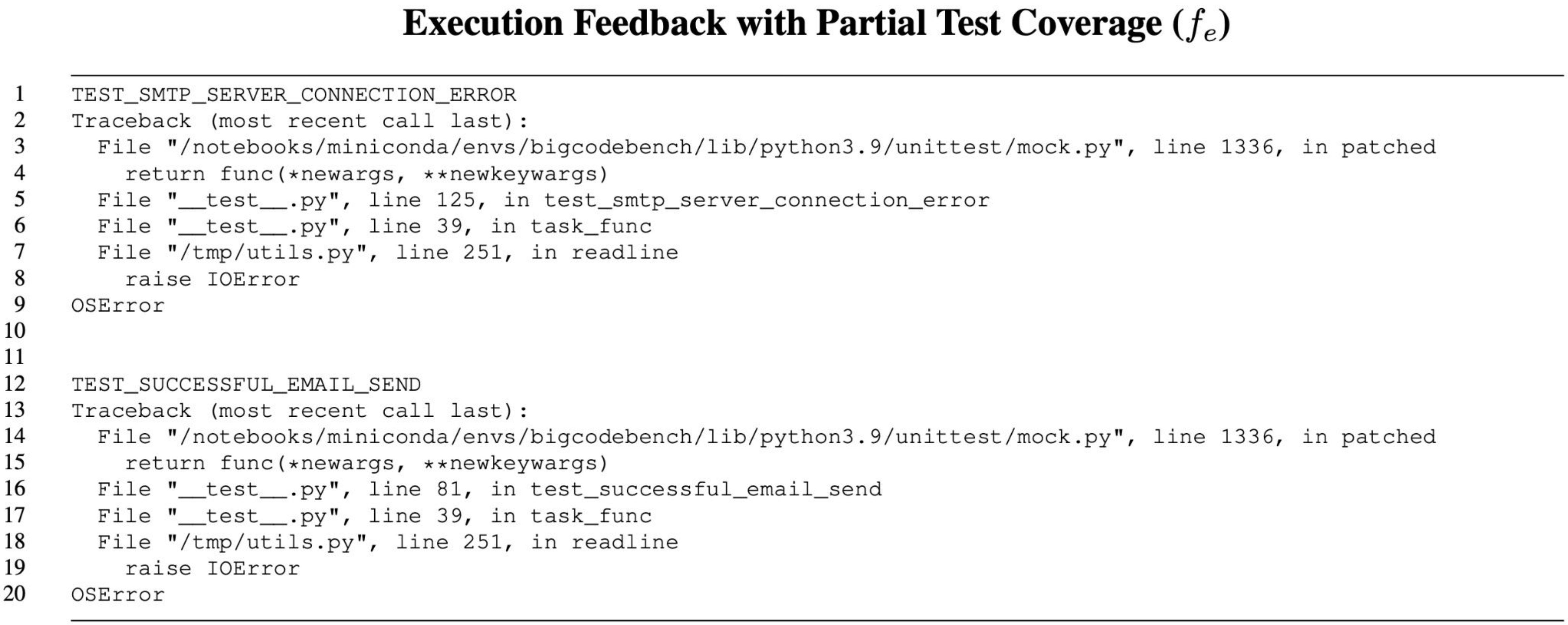
```
Compilation Feedback:
Traceback (most recent call last):
  File "tmp.py", line 6
    return int_list
                   ^
IndentationError: unindent does not match any outer indentation level
```

Types of Feedback

- Compilation Feedback (f_c)
- Execution Feedback (Test Coverage: ☰ Full | ☰ Partial) ($f_e \mid f_e^*$)

Types of Feedback

- Compilation Feedback (f_c)
- Execution Feedback (Test Coverage: ☰ Full | ☰ Partial) (f_e | f_e^*)



Execution Feedback with Partial Test Coverage (f_e)

```
TEST_SMTP_SERVER_CONNECTION_ERROR
Traceback (most recent call last):
  File "/notebooks/miniconda/envs/bigcodebench/lib/python3.9/unittest/mock.py", line 1336, in patched
    return func(*newargs, **newkeywargs)
  File "__test__.py", line 125, in test_smtp_server_connection_error
  File "__test__.py", line 39, in task_func
  File "/tmp/utils.py", line 251, in readline
    raise IOError
OSError


TEST_SUCCESSFUL_EMAIL_SEND
Traceback (most recent call last):
  File "/notebooks/miniconda/envs/bigcodebench/lib/python3.9/unittest/mock.py", line 1336, in patched
    return func(*newargs, **newkeywargs)
  File "__test__.py", line 81, in test_successful_email_send
  File "__test__.py", line 39, in task_func
  File "/tmp/utils.py", line 251, in readline
    raise IOError
OSError
```

Types of Feedback

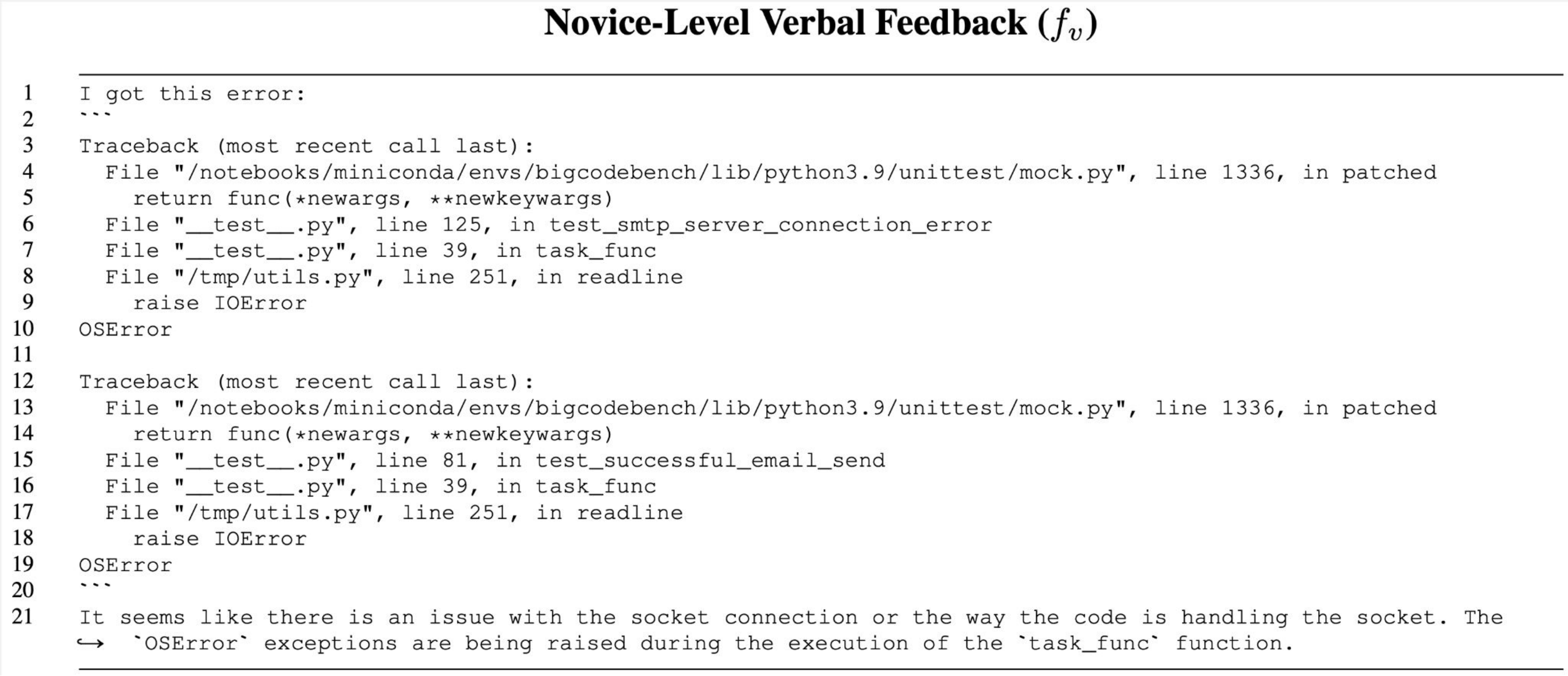
- Compilation Feedback (f_c)
- Execution Feedback (Test Coverage: ☰ Full | ☰ Partial) (f_e | f_e^*)
- Verbal Feedback (🐥 Novice | 😎 Expert) (f_v | f_v^*)

Types of Feedback

- Compilation Feedback (f_c)
- Execution Feedback (Test Coverage: Full | Partial) ($f_e \,|\, f_e^*$)
- Verbal Feedback (🐥 Novice | 😎 Expert) ($f_v \,|\, f_v^*$)

For reproducibility and consistency, we used an LLM (GPT-4o) to generate Verbal Feedback

Novice-Level Verbal Feedback



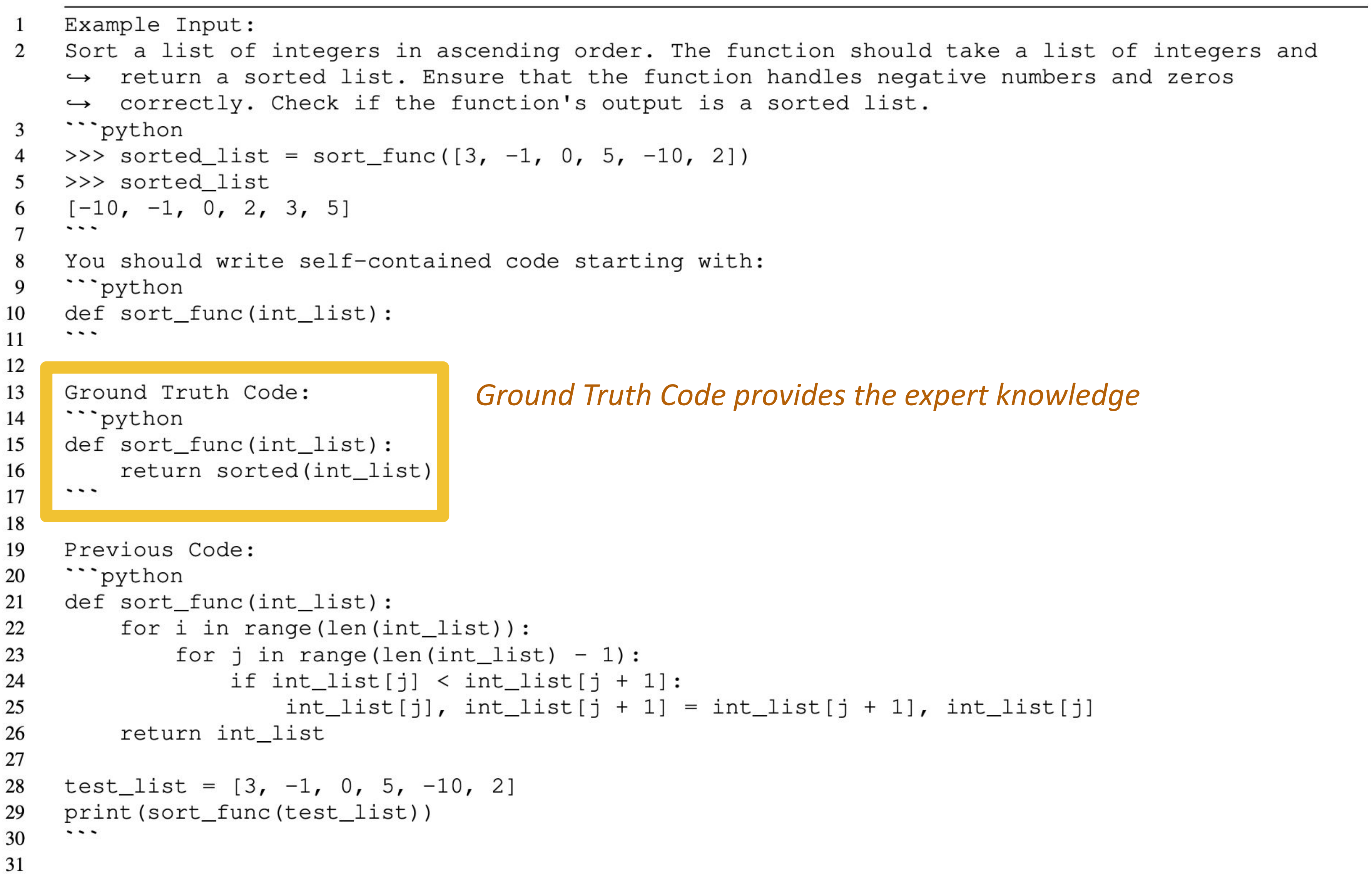
Novice-Level Verbal Feedback (f_v)

```
I got this error:
```
Traceback (most recent call last):
  File "/notebooks/miniconda/envs/bigcodebench/lib/python3.9/unittest/mock.py", line 1336, in patched
    return func(*newargs, **newkeywargs)
  File "__test__.py", line 125, in test_smtp_server_connection_error
  File "__test__.py", line 39, in task_func
  File "/tmp/utils.py", line 251, in readline
    raise IOError
OSError

Traceback (most recent call last):
  File "/notebooks/miniconda/envs/bigcodebench/lib/python3.9/unittest/mock.py", line 1336, in patched
    return func(*newargs, **newkeywargs)
  File "__test__.py", line 81, in test_successful_email_send
  File "__test__.py", line 39, in task_func
  File "/tmp/utils.py", line 251, in readline
    raise IOError
OSError
```
It seems like there is an issue with the socket connection or the way the code is handling the socket. The
↪  `OSError` exceptions are being raised during the execution of the `task_func` function.
```

Use GPT-4o to verbalize Compilation and/or Execution Feedback (+ suggestions which might be unreliable)

Expert-Level Verbal Feedback



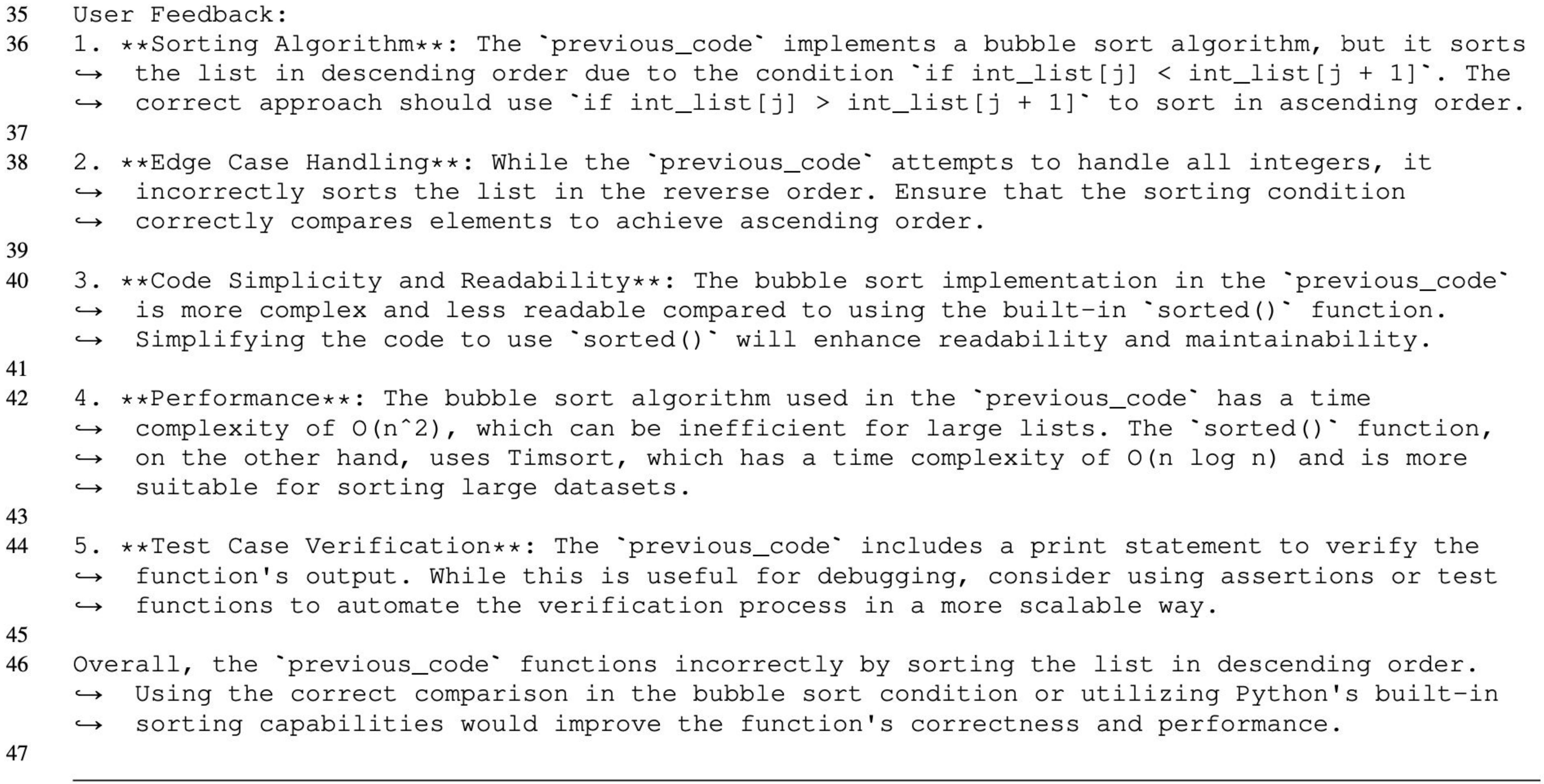
~~~
Example Input:
Sort a list of integers in ascending order. The function should take a list of integers and
↪  return a sorted list. Ensure that the function handles negative numbers and zeros
↪  correctly. Check if the function's output is a sorted list.
```python
>>> sorted_list = sort_func([3, -1, 0, 5, -10, 2])
>>> sorted_list
[-10, -1, 0, 2, 3, 5]
~~~
You should write self-contained code starting with:
```python
def sort_func(int_list):
```

Ground Truth Code:
```python
def sort_func(int_list):
    return sorted(int_list)
```

Previous Code:
```python
def sort_func(int_list):
    for i in range(len(int_list)):
        for j in range(len(int_list) - 1):
            if int_list[j] < int_list[j + 1]:
                int_list[j], int_list[j + 1] = int_list[j + 1], int_list[j]
    return int_list

test_list = [3, -1, 0, 5, -10, 2]
print(sort_func(test_list))
```

```

*Ground Truth Code provides the expert knowledge*

Input: Problem Description (+ Compilation and/or Execution Feedback) + *Ground Truth Code* + Previous Code
```

Expert-Level Verbal Feedback



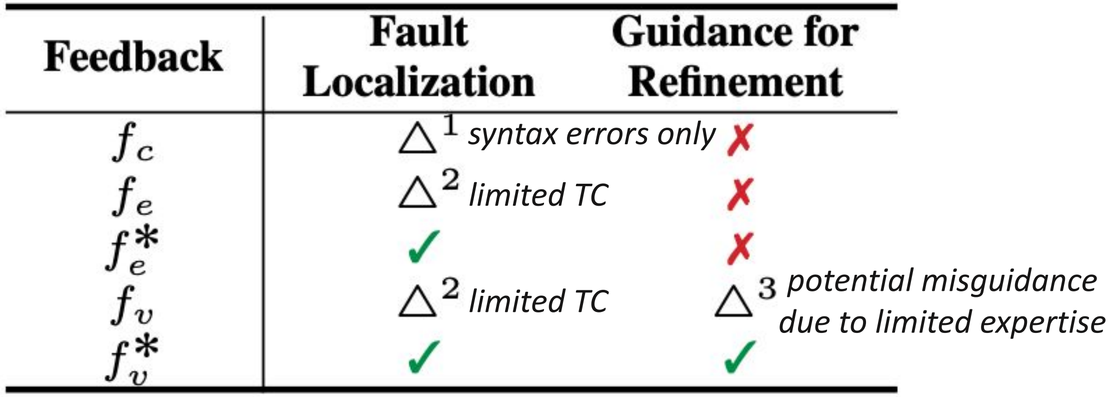
```
User Feedback:
1. **Sorting Algorithm**: The `previous_code` implements a bubble sort algorithm, but it sorts
↪  the list in descending order due to the condition `if int_list[j] < int_list[j + 1]`. The
↪  correct approach should use `if int_list[j] > int_list[j + 1]` to sort in ascending order.

2. **Edge Case Handling**: While the `previous_code` attempts to handle all integers, it
↪  incorrectly sorts the list in the reverse order. Ensure that the sorting condition
↪  correctly compares elements to achieve ascending order.

3. **Code Simplicity and Readability**: The bubble sort implementation in the `previous_code`
↪  is more complex and less readable compared to using the built-in `sorted()` function.
↪  Simplifying the code to use `sorted()` will enhance readability and maintainability.

4. **Performance**: The bubble sort algorithm used in the `previous_code` has a time
↪  complexity of O(n^2), which can be inefficient for large lists. The `sorted()` function,
↪  on the other hand, uses Timsort, which has a time complexity of O(n log n) and is more
↪  suitable for sorting large datasets.

5. **Test Case Verification**: The `previous_code` includes a print statement to verify the
↪  function's output. While this is useful for debugging, consider using assertions or test
↪  functions to automate the verification process in a more scalable way.

Overall, the `previous_code` functions incorrectly by sorting the list in descending order.
↪  Using the correct comparison in the bubble sort condition or utilizing Python's built-in
↪  sorting capabilities would improve the function's correctness and performance.
```

Types of Feedback

- Compilation Feedback (f_c)
- Execution Feedback (Test Coverage: Full | Partial) ($f_e \mid f_e^*$)
- Verbal Feedback (Novice | Expert) ($f_v \mid f_v^*$)

Types of Feedback

- Compilation Feedback (f_c)
- Execution Feedback (Test Coverage: ☰ Full | ☰ Partial) (f_e | f_e^*)
- Verbal Feedback (🐥 Novice | 😎 Expert) (f_v | f_v^*)



Feedback	Fault Localization	Guidance for Refinement
f_c	$\triangle^1$ *syntax errors only*	✗
f_e	$\triangle^2$ *limited TC*	✗
f_e^*	✓	✗
f_v	$\triangle^2$ *limited TC*	$\triangle^3$ *potential misguidance due to limited expertise*
f_v^*	✓	✓

Types of Feedback

- Compilation Feedback (f_c)
- Execution Feedback (Test Coverage: Full | Partial) ($f_e | f_e^*$)
- Verbal Feedback (Novice | Expert) ($f_v | f_v^*$)

Feedback	Fault Localization	Guidance for Refinement
f_c	$\triangle^1$ *syntax errors only*	✗
f_e	$\triangle^2$ *limited TC*	✗
f_e^*	✓	✗
f_v	$\triangle^2$ *limited TC*	$\triangle^3$ *potential misguidance due to limited expertise*
f_v^*	✓	✓

$$\Omega = \langle f_c, [\phi | f_e | f_e^*], [\phi | f_v | f_v^*] \rangle$$

Ω	InterCode	MINT	CONVCODEWORLD
$\langle f_c, \phi, \phi \rangle$	✗	✗	✓
$\langle f_c, f_e, \phi \rangle$	✗	✓	✓
$\langle f_c, f_e^*, \phi \rangle$	✓	✗	✓
$\langle f_c, \phi, f_v \rangle$	✗	✗	✓
$\langle f_c, f_e, f_v \rangle$	✗	✓	✓
$\langle f_c, f_e^*, f_v \rangle$	✗	✗	✓
$\langle f_c, \phi, f_v^* \rangle$	✗	✗	✓
$\langle f_c, f_e, f_v^* \rangle$	✗	✓	✓
$\langle f_c, f_e^*, f_v^* \rangle$	✗	✗	✓

Setup

Base Benchmark: BigCodeBench-Full-Instruct
- highly challenging problem sets (Prev. SOTA: 51.1 of Pass@1)
- large scale (1,140 problems)

Maximum Number of Iterations: 10

Evaluation Metrics
- Mean Reciprocal Rank (**MRR**): 1/k where k is the turn at which the model produces correct code
  → Measures how fast correct code is returned.
- **Recall**: 1 if the model produces correct code within n turns.
  → Measures whether correct code is eventually returned.

Experimental Results

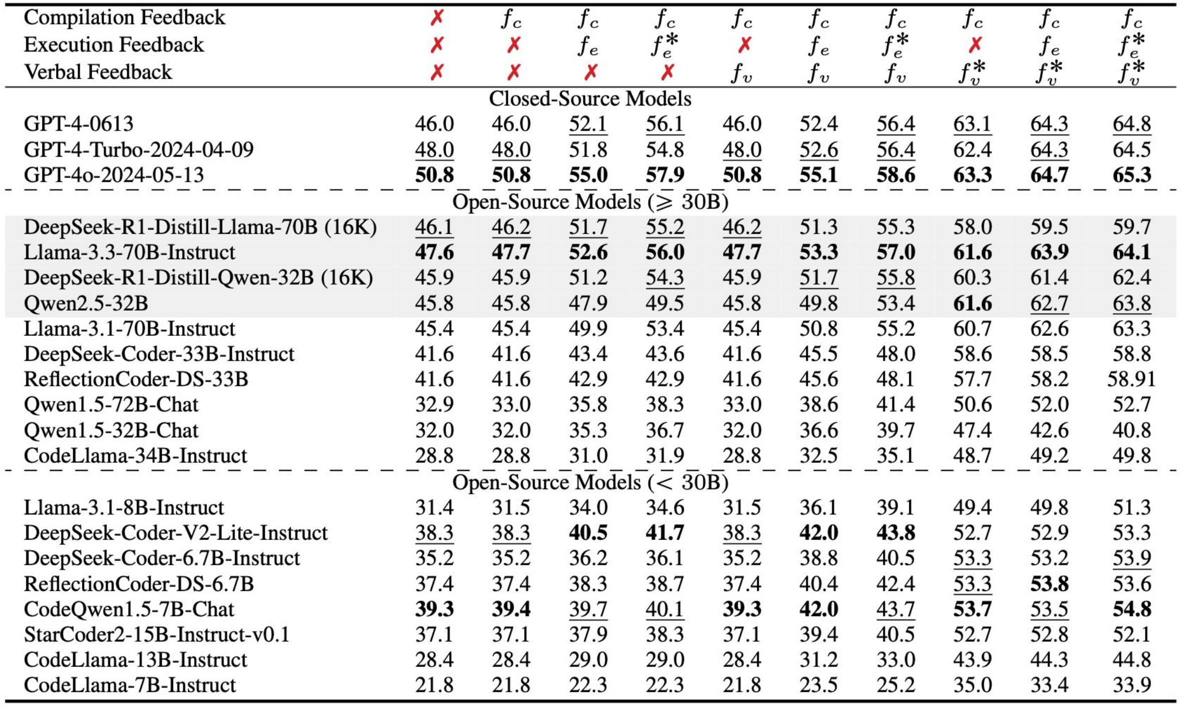


Table 7: MRR results on CONVCODEWORLD. ✗ indicates that no feedback of that type is provided (ϕ). The leftmost results, with three ✗, represent $\Omega = \langle \phi, \phi, \phi \rangle$, corresponding to single-turn code generation without any feedback. For each column, bold and underscore indicate 1st and 2nd place performance within the same model group. Maximum token length is set to 8K throughout the experiments, except for the R1-Distill models, which are set to 16K.

Compilation Feedback	✗	f_c	f_c	f_c	f_c	f_c	f_c	f_c	f_c	f_c
Execution Feedback	✗	✗	f_e	f_e^*	✗	f_e	f_e^*	✗	f_e	f_e^*
Verbal Feedback	✗	✗	✗	✗	f_v	f_v	f_v	f_v^*	f_v^*	f_v^*
Closed-Source Models										
GPT-4-0613	46.0	46.0	52.1	56.1	46.0	52.4	56.4	63.1	64.3	64.8
GPT-4-Turbo-2024-04-09	48.0	48.0	51.8	54.8	48.0	52.6	56.4	62.4	64.3	64.5
GPT-4o-2024-05-13	**50.8**	**50.8**	**55.0**	**57.9**	**50.8**	**55.1**	**58.6**	**63.3**	**64.7**	**65.3**
Open-Source Models ($\geqslant$ 30B)										
DeepSeek-R1-Distill-Llama-70B (16K)	46.1	46.2	51.7	55.2	46.2	51.3	55.3	58.0	59.5	59.7
Llama-3.3-70B-Instruct	**47.6**	**47.7**	**52.6**	**56.0**	**47.7**	**53.3**	**57.0**	**61.6**	**63.9**	**64.1**
DeepSeek-R1-Distill-Qwen-32B (16K)	45.9	45.9	51.2	54.3	45.9	51.7	55.8	60.3	61.4	62.4
Qwen2.5-32B	45.8	45.8	47.9	49.5	45.8	49.8	53.4	**61.6**	62.7	63.8
Llama-3.1-70B-Instruct	45.4	45.4	49.9	53.4	45.4	50.8	55.2	60.7	62.6	63.3
DeepSeek-Coder-33B-Instruct	41.6	41.6	43.4	43.6	41.6	45.5	48.0	58.6	58.5	58.8
ReflectionCoder-DS-33B	41.6	41.6	42.9	42.9	41.6	45.6	48.1	57.7	58.2	58.91
Qwen1.5-72B-Chat	32.9	33.0	35.8	38.3	33.0	38.6	41.4	50.6	52.0	52.7
Qwen1.5-32B-Chat	32.0	32.0	35.3	36.7	32.0	36.6	39.7	47.4	42.6	40.8
CodeLlama-34B-Instruct	28.8	28.8	31.0	31.9	28.8	32.5	35.1	48.7	49.2	49.8
Open-Source Models ($<$ 30B)										
Llama-3.1-8B-Instruct	31.4	31.5	34.0	34.6	31.5	36.1	39.1	49.4	49.8	51.3
DeepSeek-Coder-V2-Lite-Instruct	38.3	38.3	**40.5**	**41.7**	38.3	**42.0**	**43.8**	52.7	52.9	53.3
DeepSeek-Coder-6.7B-Instruct	35.2	35.2	36.2	36.1	35.2	38.8	40.5	53.3	53.2	53.9
ReflectionCoder-DS-6.7B	37.4	37.4	38.3	38.7	37.4	40.4	42.4	53.3	**53.8**	53.6
CodeQwen1.5-7B-Chat	**39.3**	**39.4**	39.7	40.1	**39.3**	**42.0**	43.7	**53.7**	53.5	**54.8**
StarCoder2-15B-Instruct-v0.1	37.1	37.1	37.9	38.3	37.1	39.4	40.5	52.7	52.8	52.1
CodeLlama-13B-Instruct	28.4	28.4	29.0	29.0	28.4	31.2	33.0	43.9	44.3	44.8
CodeLlama-7B-Instruct	21.8	21.8	22.3	22.3	21.8	23.5	25.2	35.0	33.4	33.9

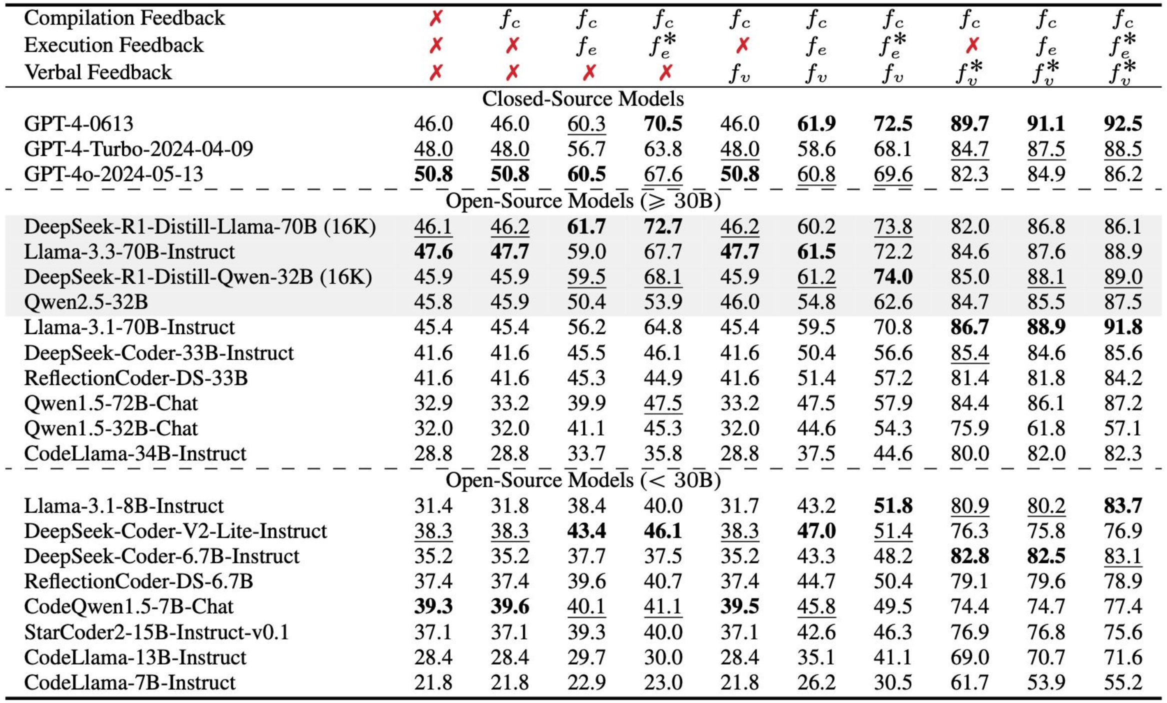


Table 8: Recall results on CONVCODEWORLD. ✗ indicates that no feedback of that type is provided (ϕ). The leftmost results, with three ✗, represent $\Omega = \langle \phi, \phi, \phi \rangle$, corresponding to single-turn code generation without any feedback. For each column, bold and underscore indicate 1st and 2nd place performance within the same model group. Maximum token length is set to 8K throughout the experiments, except for the R1-Distill models, which are set to 16K.

Compilation Feedback	✗	f_c	f_c	f_c	f_c	f_c	f_c	f_c	f_c	f_c
Execution Feedback	✗	✗	f_e	f_e^*	✗	f_e	f_e^*	✗	f_e	f_e^*
Verbal Feedback	✗	✗	✗	✗	f_v	f_v	f_v	f_v^*	f_v^*	f_v^*
Closed-Source Models										
GPT-4-0613	46.0	46.0	60.3	**70.5**	46.0	**61.9**	**72.5**	**89.7**	**91.1**	**92.5**
GPT-4-Turbo-2024-04-09	48.0	48.0	56.7	63.8	48.0	58.6	68.1	84.7	87.5	88.5
GPT-4o-2024-05-13	**50.8**	**50.8**	**60.5**	67.6	**50.8**	60.8	69.6	82.3	84.9	86.2
Open-Source Models ($\geqslant$ 30B)										
DeepSeek-R1-Distill-Llama-70B (16K)	46.1	46.2	**61.7**	**72.7**	46.2	60.2	73.8	82.0	86.8	86.1
Llama-3.3-70B-Instruct	**47.6**	**47.7**	59.0	67.7	**47.7**	**61.5**	72.2	84.6	87.6	88.9
DeepSeek-R1-Distill-Qwen-32B (16K)	45.9	45.9	59.5	68.1	45.9	61.2	**74.0**	85.0	88.1	89.0
Qwen2.5-32B	45.8	45.9	50.4	53.9	46.0	54.8	62.6	84.7	85.5	87.5
Llama-3.1-70B-Instruct	45.4	45.4	56.2	64.8	45.4	59.5	70.8	**86.7**	**88.9**	**91.8**
DeepSeek-Coder-33B-Instruct	41.6	41.6	45.5	46.1	41.6	50.4	56.6	85.4	84.6	85.6
ReflectionCoder-DS-33B	41.6	41.6	45.3	44.9	41.6	51.4	57.2	81.4	81.8	84.2
Qwen1.5-72B-Chat	32.9	33.2	39.9	47.5	33.2	47.5	57.9	84.4	86.1	87.2
Qwen1.5-32B-Chat	32.0	32.0	41.1	45.3	32.0	44.6	54.3	75.9	61.8	57.1
CodeLlama-34B-Instruct	28.8	28.8	33.7	35.8	28.8	37.5	44.6	80.0	82.0	82.3
Open-Source Models ($<$ 30B)										
Llama-3.1-8B-Instruct	31.4	31.8	38.4	40.0	31.7	43.2	**51.8**	80.9	80.2	**83.7**
DeepSeek-Coder-V2-Lite-Instruct	38.3	38.3	**43.4**	**46.1**	38.3	**47.0**	51.4	76.3	75.8	76.9
DeepSeek-Coder-6.7B-Instruct	35.2	35.2	37.7	37.5	35.2	43.3	48.2	**82.8**	**82.5**	83.1
ReflectionCoder-DS-6.7B	37.4	37.4	39.6	40.7	37.4	44.7	50.4	79.1	79.6	78.9
CodeQwen1.5-7B-Chat	**39.3**	**39.6**	40.1	41.1	**39.5**	45.8	49.5	74.4	74.7	77.4
StarCoder2-15B-Instruct-v0.1	37.1	37.1	39.3	40.0	37.1	42.6	46.3	76.9	76.8	75.6
CodeLlama-13B-Instruct	28.4	28.4	29.7	30.0	28.4	35.1	41.1	69.0	70.7	71.6
CodeLlama-7B-Instruct	21.8	21.8	22.9	23.0	21.8	26.2	30.5	61.7	53.9	55.2

Experimental Results

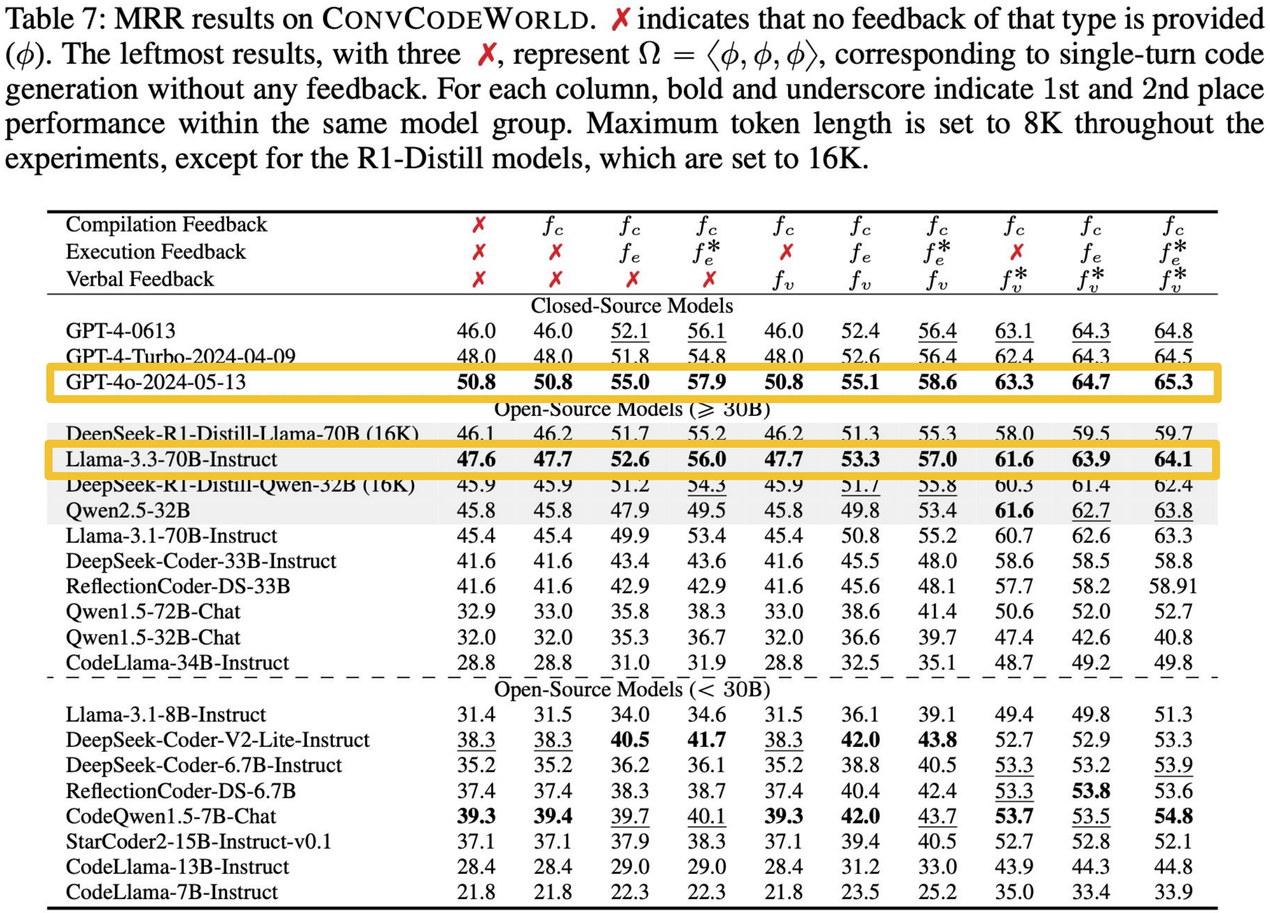


Table 7: MRR results on ConvCodeWorld. ✗ indicates that no feedback of that type is provided (ϕ). The leftmost results, with three ✗, represent $\Omega = \langle \phi, \phi, \phi \rangle$, corresponding to single-turn code generation without any feedback. For each column, bold and underscore indicate 1st and 2nd place performance within the same model group. Maximum token length is set to 8K throughout the experiments, except for the R1-Distill models, which are set to 16K.

Compilation Feedback Execution Feedback Verbal Feedback	✗ ✗ ✗	f_c ✗ ✗	f_c f_e ✗	f_c f_e^* ✗	f_c ✗ f_v	f_c f_e f_v	f_c f_e^* f_v	f_c ✗ f_v^*	f_c f_e f_v^*	f_c f_e^* f_v^*
Closed-Source Models										
GPT-4-0613	46.0	46.0	52.1	56.1	46.0	52.4	56.4	63.1	64.3	64.8
GPT-4-Turbo-2024-04-09	48.0	48.0	51.8	54.8	48.0	52.6	56.4	62.4	64.3	64.5
GPT-4o-2024-05-13	**50.8**	**50.8**	**55.0**	**57.9**	**50.8**	**55.1**	**58.6**	**63.3**	**64.7**	**65.3**
Open-Source Models ($\geqslant$ 30B)										
DeepSeek-R1-Distill-Llama-70B (16K)	46.1	46.2	51.7	55.2	46.2	51.3	55.3	58.0	59.5	59.7
Llama-3.3-70B-Instruct	**47.6**	**47.7**	**52.6**	**56.0**	**47.7**	**53.3**	**57.0**	**61.6**	**63.9**	**64.1**
DeepSeek-R1-Distill-Qwen-32B (16K)	45.9	45.9	51.2	54.3	45.9	51.7	55.8	60.3	61.4	62.4
Qwen2.5-32B	45.8	45.8	47.9	49.5	45.8	49.8	53.4	**61.6**	62.7	63.8
Llama-3.1-70B-Instruct	45.4	45.4	49.9	53.4	45.4	50.8	55.2	60.7	62.6	63.3
DeepSeek-Coder-33B-Instruct	41.6	41.6	43.4	43.6	41.6	45.5	48.0	58.6	58.5	58.8
ReflectionCoder-DS-33B	41.6	41.6	42.9	42.9	41.6	45.6	48.1	57.7	58.2	58.91
Qwen1.5-72B-Chat	32.9	33.0	35.8	38.3	33.0	38.6	41.4	50.6	52.0	52.7
Qwen1.5-32B-Chat	32.0	32.0	35.3	36.7	32.0	36.6	39.7	47.4	42.6	40.8
CodeLlama-34B-Instruct	28.8	28.8	31.0	31.9	28.8	32.5	35.1	48.7	49.2	49.8
Open-Source Models (< 30B)										
Llama-3.1-8B-Instruct	31.4	31.5	34.0	34.6	31.5	36.1	39.1	49.4	49.8	51.3
DeepSeek-Coder-V2-Lite-Instruct	38.3	38.3	**40.5**	**41.7**	38.3	**42.0**	**43.8**	52.7	52.9	53.3
DeepSeek-Coder-6.7B-Instruct	35.2	35.2	36.2	36.1	35.2	38.8	40.5	53.3	53.2	53.9
ReflectionCoder-DS-6.7B	37.4	37.4	38.3	38.7	37.4	40.4	42.4	53.3	**53.8**	53.6
CodeQwen1.5-7B-Chat	**39.3**	**39.4**	39.7	40.1	**39.3**	**42.0**	43.7	**53.7**	53.5	**54.8**
StarCoder2-15B-Instruct-v0.1	37.1	37.1	37.9	38.3	37.1	39.4	40.5	52.7	52.8	52.1
CodeLlama-13B-Instruct	28.4	28.4	29.0	29.0	28.4	31.2	33.0	43.9	44.3	44.8
CodeLlama-7B-Instruct	21.8	21.8	22.3	22.3	21.8	23.5	25.2	35.0	33.4	33.9

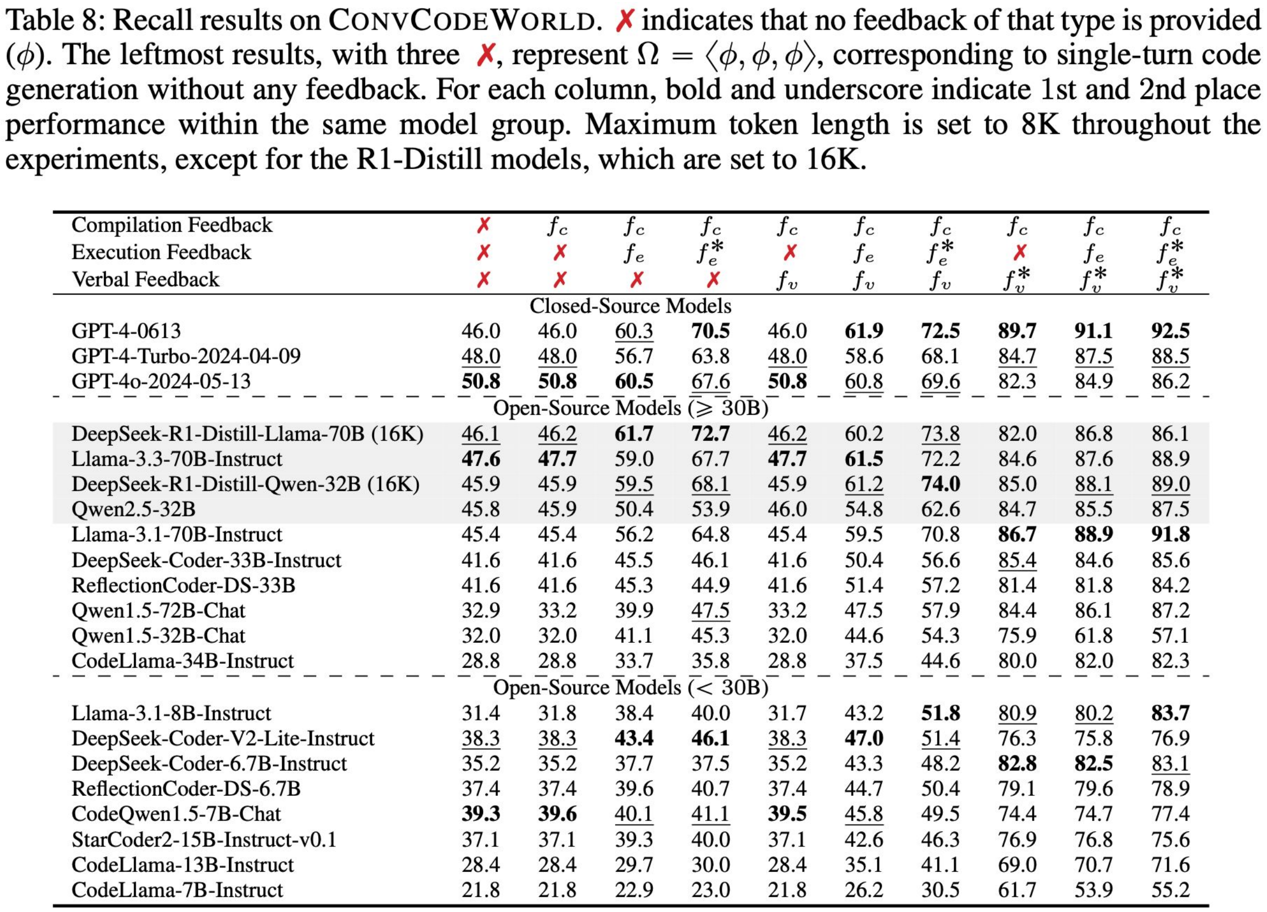


Table 8: Recall results on ConvCodeWorld. ✗ indicates that no feedback of that type is provided (ϕ). The leftmost results, with three ✗, represent $\Omega = \langle \phi, \phi, \phi \rangle$, corresponding to single-turn code generation without any feedback. For each column, bold and underscore indicate 1st and 2nd place performance within the same model group. Maximum token length is set to 8K throughout the experiments, except for the R1-Distill models, which are set to 16K.

Compilation Feedback Execution Feedback Verbal Feedback	✗ ✗ ✗	f_c ✗ ✗	f_c f_e ✗	f_c f_e^* ✗	f_c ✗ f_v	f_c f_e f_v	f_c f_e^* f_v	f_c ✗ f_v^*	f_c f_e f_v^*	f_c f_e^* f_v^*
Closed-Source Models										
GPT-4-0613	46.0	46.0	60.3	**70.5**	46.0	**61.9**	**72.5**	**89.7**	**91.1**	**92.5**
GPT-4-Turbo-2024-04-09	48.0	48.0	56.7	63.8	48.0	58.6	68.1	84.7	87.5	88.5
GPT-4o-2024-05-13	**50.8**	**50.8**	**60.5**	67.6	**50.8**	60.8	69.6	82.3	84.9	86.2
Open-Source Models ($\geqslant$ 30B)										
DeepSeek-R1-Distill-Llama-70B (16K)	46.1	46.2	**61.7**	**72.7**	46.2	60.2	73.8	82.0	86.8	86.1
Llama-3.3-70B-Instruct	**47.6**	**47.7**	59.0	67.7	**47.7**	**61.5**	72.2	84.6	87.6	88.9
DeepSeek-R1-Distill-Qwen-32B (16K)	45.9	45.9	59.5	68.1	45.9	61.2	**74.0**	85.0	88.1	89.0
Qwen2.5-32B	45.8	45.9	50.4	53.9	46.0	54.8	62.6	84.7	85.5	87.5
Llama-3.1-70B-Instruct	45.4	45.4	56.2	64.8	45.4	59.5	70.8	**86.7**	**88.9**	**91.8**
DeepSeek-Coder-33B-Instruct	41.6	41.6	45.5	46.1	41.6	50.4	56.6	85.4	84.6	85.6
ReflectionCoder-DS-33B	41.6	41.6	45.3	44.9	41.6	51.4	57.2	81.4	81.8	84.2
Qwen1.5-72B-Chat	32.9	33.2	39.9	47.5	33.2	47.5	57.9	84.4	86.1	87.2
Qwen1.5-32B-Chat	32.0	32.0	41.1	45.3	32.0	44.6	54.3	75.9	61.8	57.1
CodeLlama-34B-Instruct	28.8	28.8	33.7	35.8	28.8	37.5	44.6	80.0	82.0	82.3
Open-Source Models (< 30B)										
Llama-3.1-8B-Instruct	31.4	31.8	38.4	40.0	31.7	43.2	**51.8**	80.9	80.2	**83.7**
DeepSeek-Coder-V2-Lite-Instruct	38.3	38.3	**43.4**	**46.1**	38.3	**47.0**	51.4	76.3	75.8	76.9
DeepSeek-Coder-6.7B-Instruct	35.2	35.2	37.7	37.5	35.2	43.3	48.2	**82.8**	**82.5**	83.1
ReflectionCoder-DS-6.7B	37.4	37.4	39.6	40.7	37.4	44.7	50.4	79.1	79.6	78.9
CodeQwen1.5-7B-Chat	**39.3**	**39.6**	40.1	41.1	**39.5**	45.8	49.5	74.4	74.7	77.4
StarCoder2-15B-Instruct-v0.1	37.1	37.1	39.3	40.0	37.1	42.6	46.3	76.9	76.8	75.6
CodeLlama-13B-Instruct	28.4	28.4	29.7	30.0	28.4	35.1	41.1	69.0	70.7	71.6
CodeLlama-7B-Instruct	21.8	21.8	22.9	23.0	21.8	26.2	30.5	61.7	53.9	55.2

Trend difference between MRR and Recall

- MRR of Closed-Source and Open-Source (>= 30B): **Clear winner exists**
- Recall: No clear winners

Experimental Results

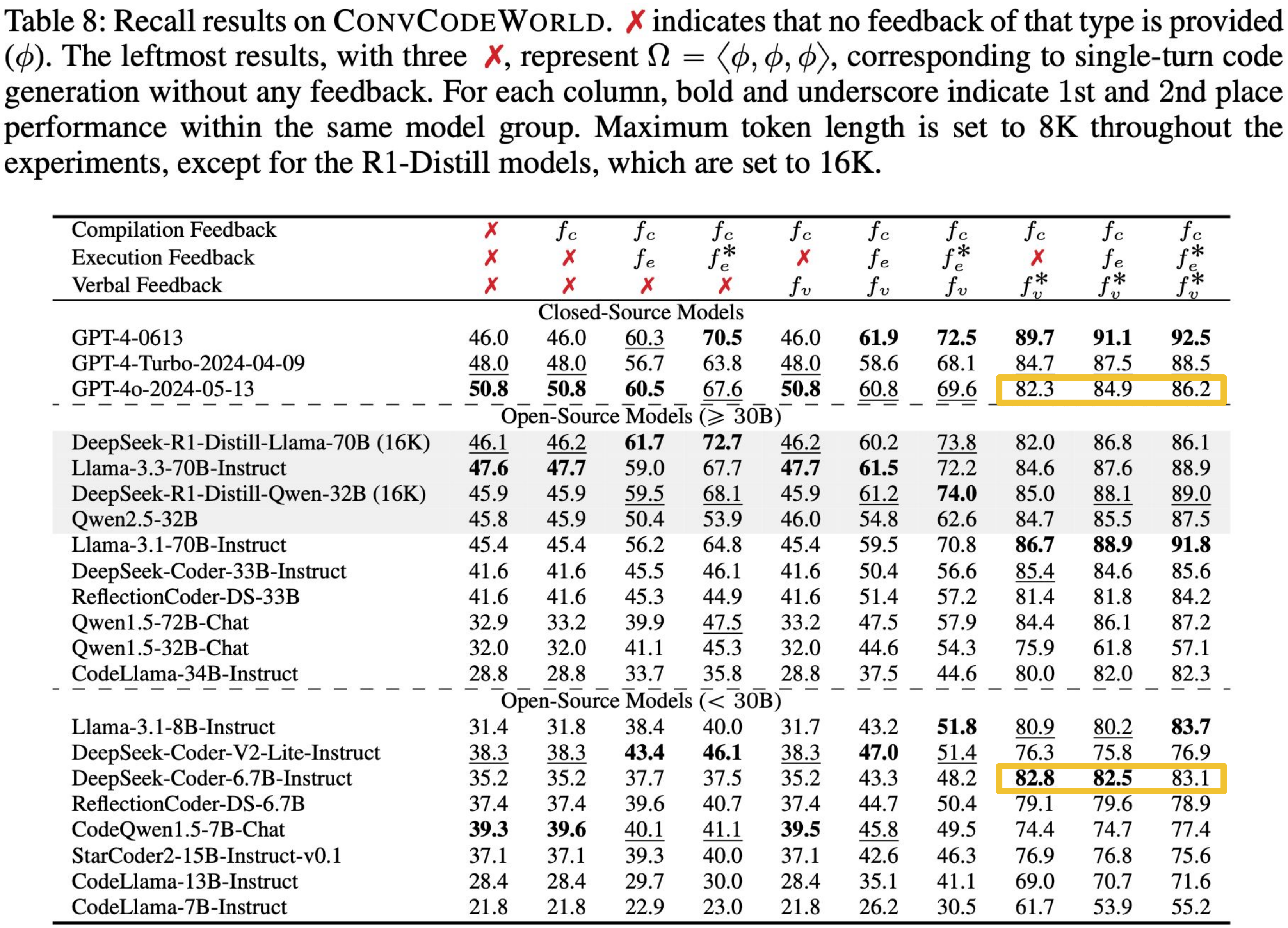
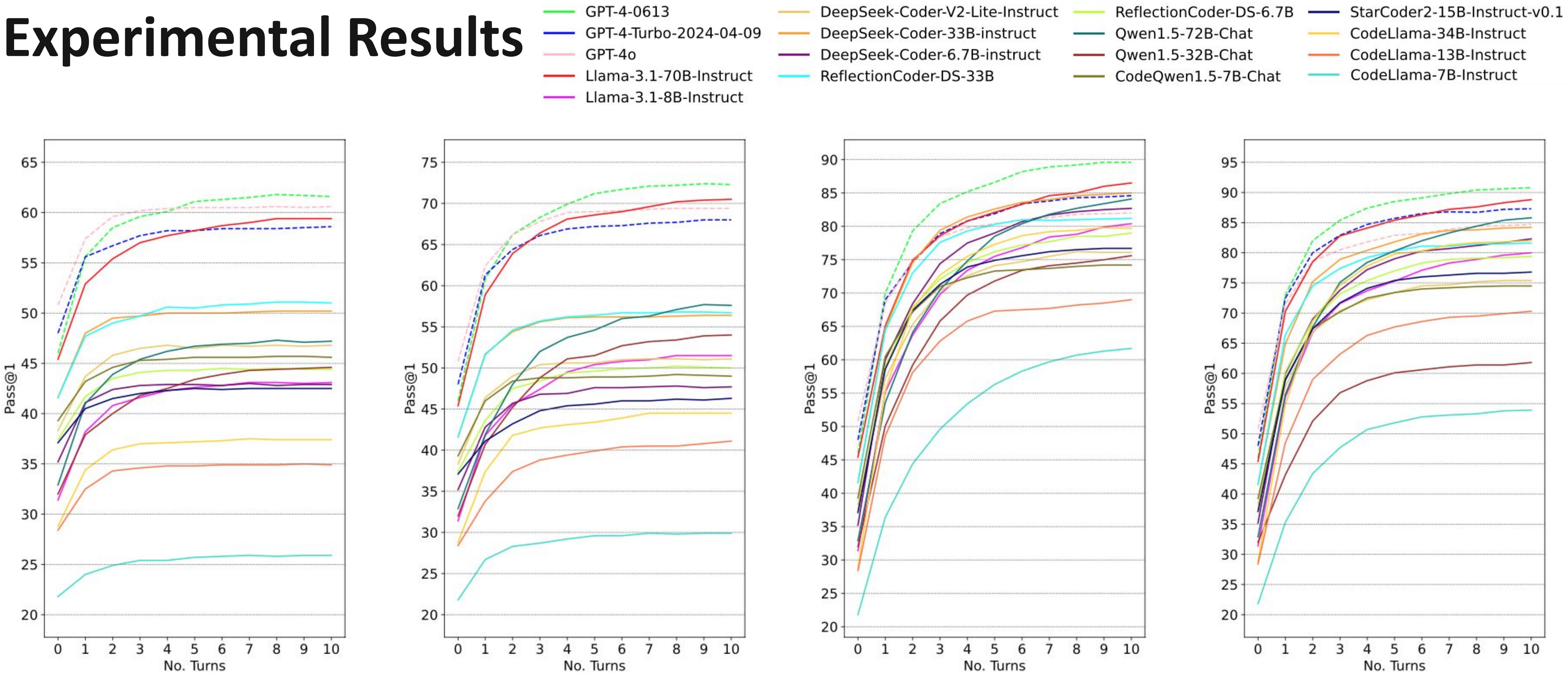


Table 8: Recall results on CONVCODEWORLD. ✗ indicates that no feedback of that type is provided (ϕ). The leftmost results, with three ✗, represent $\Omega = \langle \phi, \phi, \phi \rangle$, corresponding to single-turn code generation without any feedback. For each column, bold and underscore indicate 1st and 2nd place performance within the same model group. Maximum token length is set to 8K throughout the experiments, except for the R1-Distill models, which are set to 16K.

Compilation Feedback	✗	f_c	f_c	f_c	f_c	f_c	f_c	f_c	f_c	f_c
Execution Feedback	✗	✗	f_e	f_e^*	✗	f_e	f_e^*	✗	f_e	f_e^*
Verbal Feedback	✗	✗	✗	✗	f_v	f_v	f_v	f_v^*	f_v^*	f_v^*
Closed-Source Models										
GPT-4-0613	46.0	46.0	60.3	**70.5**	46.0	**61.9**	**72.5**	**89.7**	**91.1**	**92.5**
GPT-4-Turbo-2024-04-09	48.0	48.0	56.7	63.8	48.0	58.6	68.1	84.7	87.5	88.5
GPT-4o-2024-05-13	**50.8**	**50.8**	**60.5**	67.6	**50.8**	60.8	69.6	82.3	84.9	86.2
Open-Source Models ($\geqslant$ 30B)										
DeepSeek-R1-Distill-Llama-70B (16K)	46.1	46.2	**61.7**	**72.7**	46.2	60.2	73.8	82.0	86.8	86.1
Llama-3.3-70B-Instruct	**47.6**	**47.7**	59.0	67.7	**47.7**	**61.5**	72.2	84.6	87.6	88.9
DeepSeek-R1-Distill-Qwen-32B (16K)	45.9	45.9	59.5	68.1	45.9	61.2	**74.0**	85.0	88.1	89.0
Qwen2.5-32B	45.8	45.9	50.4	53.9	46.0	54.8	62.6	84.7	85.5	87.5
Llama-3.1-70B-Instruct	45.4	45.4	56.2	64.8	45.4	59.5	70.8	**86.7**	**88.9**	**91.8**
DeepSeek-Coder-33B-Instruct	41.6	41.6	45.5	46.1	41.6	50.4	56.6	85.4	84.6	85.6
ReflectionCoder-DS-33B	41.6	41.6	45.3	44.9	41.6	51.4	57.2	81.4	81.8	84.2
Qwen1.5-72B-Chat	32.9	33.2	39.9	47.5	33.2	47.5	57.9	84.4	86.1	87.2
Qwen1.5-32B-Chat	32.0	32.0	41.1	45.3	32.0	44.6	54.3	75.9	61.8	57.1
CodeLlama-34B-Instruct	28.8	28.8	33.7	35.8	28.8	37.5	44.6	80.0	82.0	82.3
Open-Source Models (< 30B)										
Llama-3.1-8B-Instruct	31.4	31.8	38.4	40.0	31.7	43.2	**51.8**	80.9	80.2	**83.7**
DeepSeek-Coder-V2-Lite-Instruct	38.3	38.3	**43.4**	**46.1**	38.3	**47.0**	51.4	76.3	75.8	76.9
DeepSeek-Coder-6.7B-Instruct	35.2	35.2	37.7	37.5	35.2	43.3	48.2	**82.8**	**82.5**	83.1
ReflectionCoder-DS-6.7B	37.4	37.4	39.6	40.7	37.4	44.7	50.4	79.1	79.6	78.9
CodeQwen1.5-7B-Chat	**39.3**	**39.6**	40.1	41.1	**39.5**	45.8	49.5	74.4	74.7	77.4
StarCoder2-15B-Instruct-v0.1	37.1	37.1	39.3	40.0	37.1	42.6	46.3	76.9	76.8	75.6
CodeLlama-13B-Instruct	28.4	28.4	29.7	30.0	28.4	35.1	41.1	69.0	70.7	71.6
CodeLlama-7B-Instruct	21.8	21.8	22.9	23.0	21.8	26.2	30.5	61.7	53.9	55.2

With Expert-level Feedback, even DeepSeek-Coder-6.7B-Instruct is comparable to GPT-4o-2024-05-13

Experimental Results



(e) $\Omega = \langle f_c, f_e, f_v \rangle$

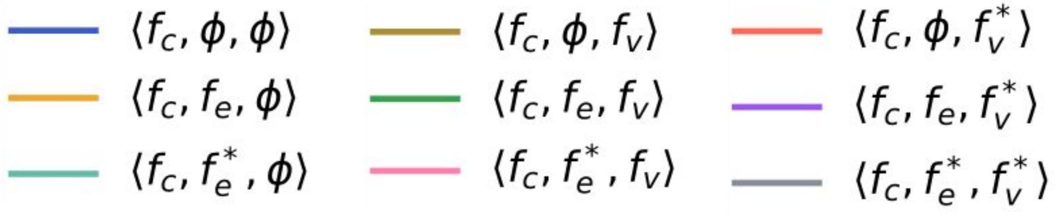
(f) $\Omega = \langle f_c, f_e^*, f_v \rangle$

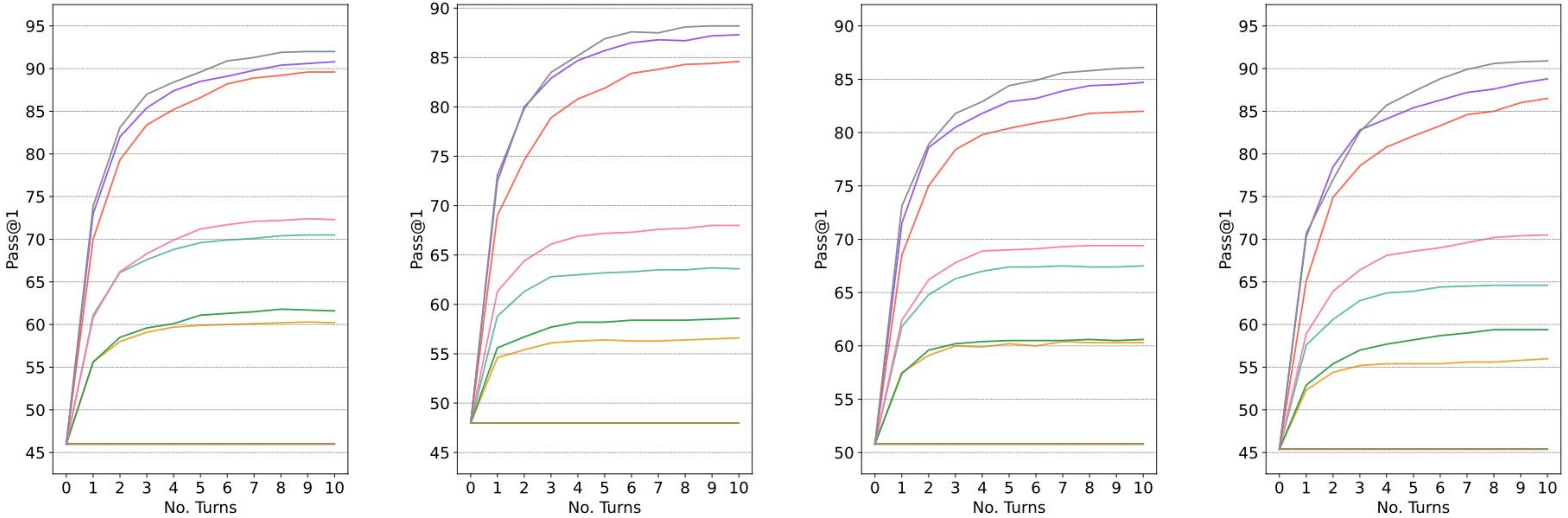
(g) $\Omega = \langle f_c, \phi, f_v^* \rangle$

(h) $\Omega = \langle f_c, f_e, f_v^* \rangle$

Weaker LLMs, with sufficient feedback, can outperform single-turn results of state-of-the-art LLMs without feedback.

Experimental Results


$\langle f_c, \phi, \phi \rangle$
$\langle f_c, \phi, f_v \rangle$
$\langle f_c, \phi, f_v^* \rangle$
$\langle f_c, f_e, \phi \rangle$
$\langle f_c, f_e, f_v \rangle$
$\langle f_c, f_e, f_v^* \rangle$
$\langle f_c, f_e^*, \phi \rangle$
$\langle f_c, f_e^*, f_v \rangle$
$\langle f_c, f_e^*, f_v^* \rangle$



(a) GPT-4-0613

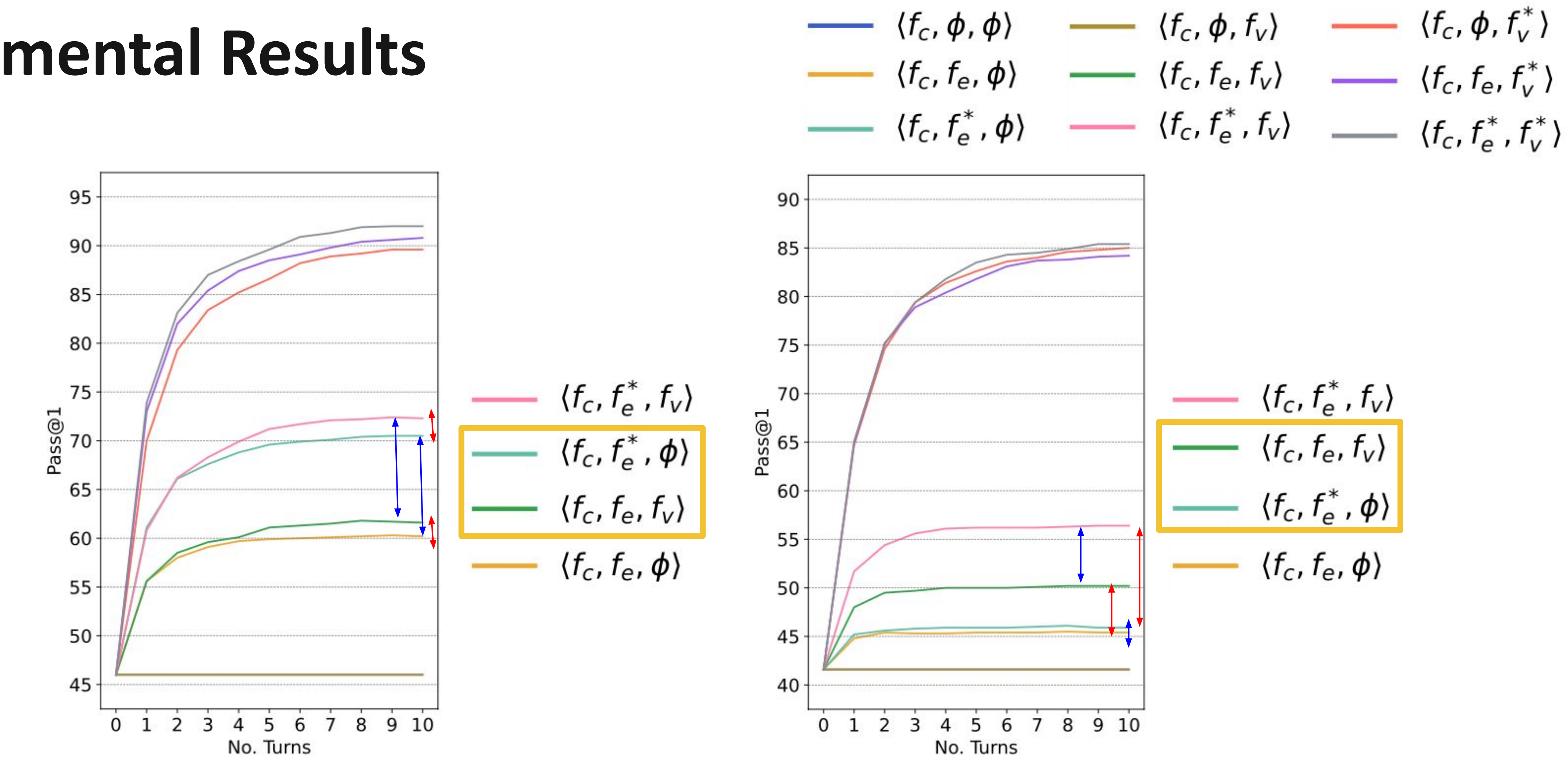
(b) GPT-4-Turbo-2024-04-09

(c) GPT-4o-2024-05-13

(d) Llama-3.1-70B-Instruct

LLM performance varies significantly based on the feedback provided.

Experimental Results



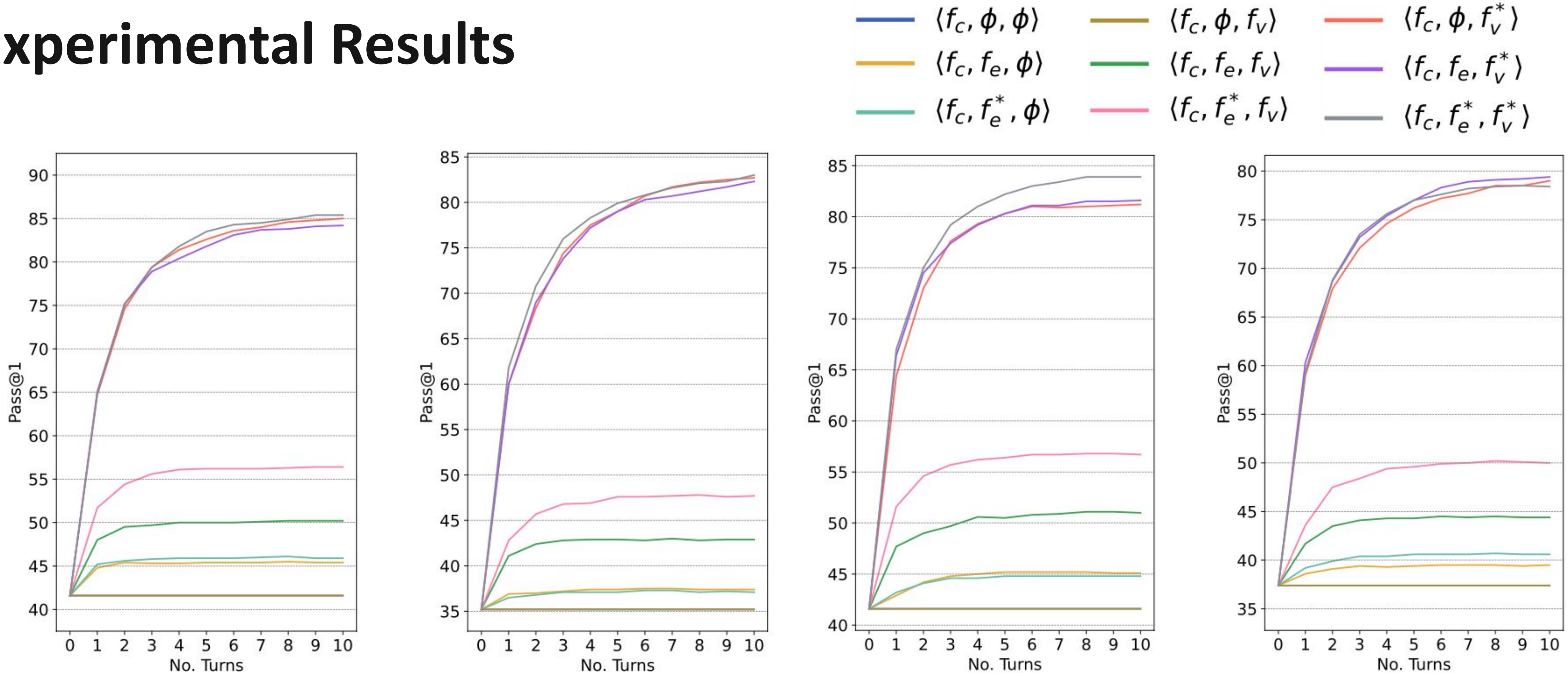
(a) GPT-4-0613

(g) DeepSeek-Coder-33B-Instruct

Adding novice-level feedback (f_v) is more effective to open-sourced models.

Improving test coverage for execution feedback (f_e^*) is more effective to closed-source models.

Experimental Results



(g) DeepSeek-Coder-33B-Instruct

(h) DeepSeek-Coder-6.7B-Instruct

(i) ReflectionCoder-DS-33B

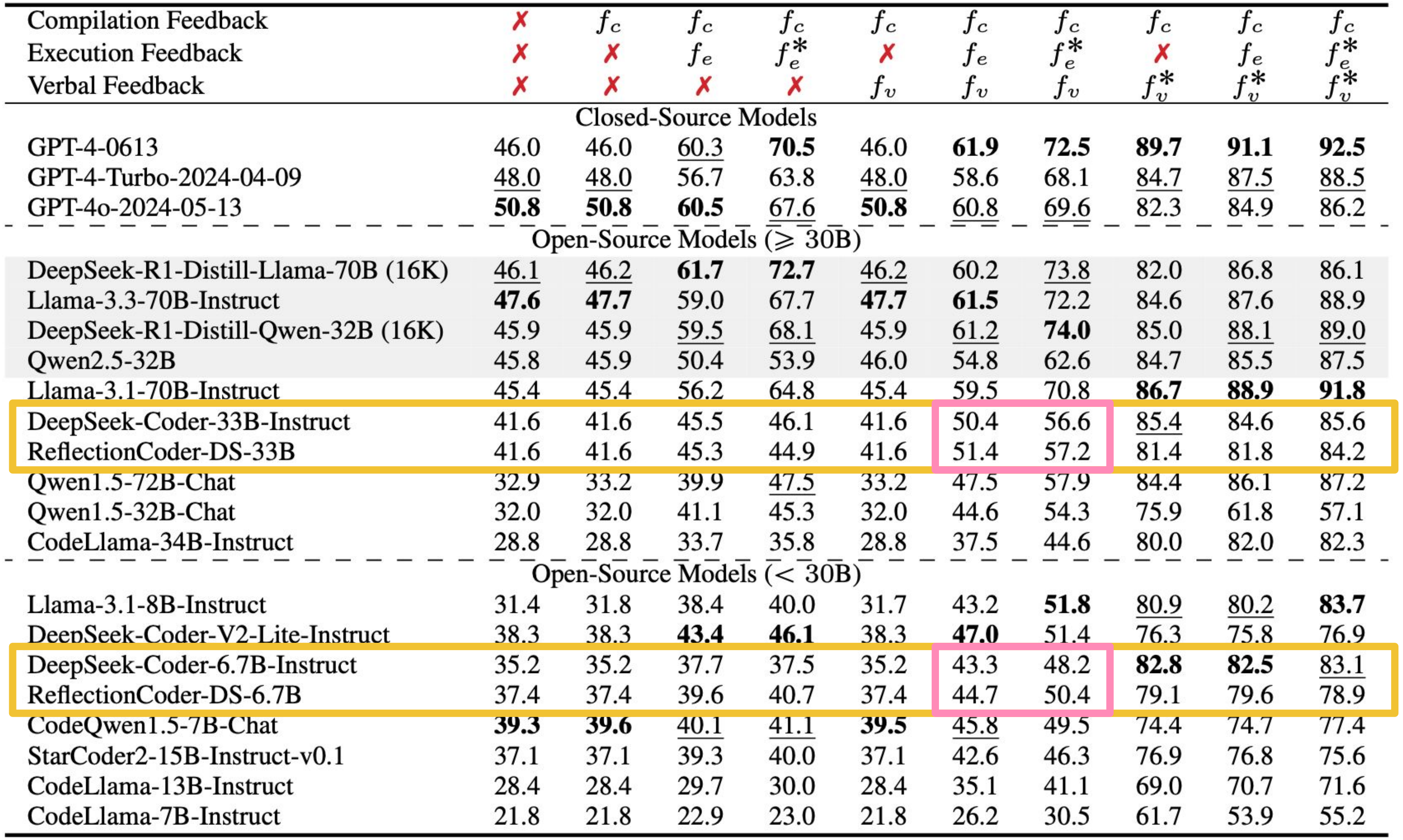
(j) ReflectionCoder-DS-6.7B

ReflectionCoder-DS: DeepSeek-Coder + SFT on $\boxed{\langle f_c, f_e^*, f_v \rangle}$

Training on a specific feedback combination can limit an LLM’s ability to utilize unseen combinations.

Experimental Results

Table 8: Recall results on CONVCODEWORLD. ✗ indicates that no feedback of that type is provided (ϕ). The leftmost results, with three ✗, represent $\Omega = \langle \phi, \phi, \phi \rangle$, corresponding to single-turn code generation without any feedback. For each column, bold and underscore indicate 1st and 2nd place performance within the same model group. Maximum token length is set to 8K throughout the experiments, except for the R1-Distill models, which are set to 16K.

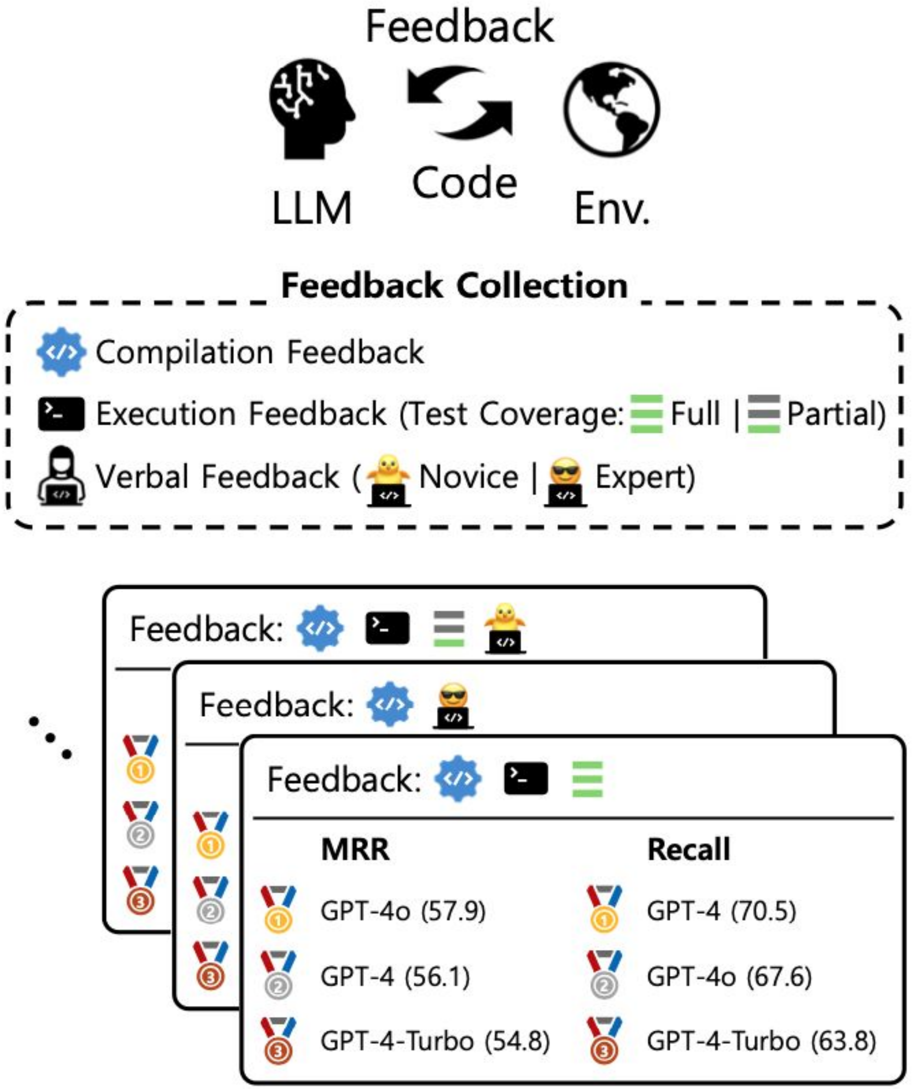


Compilation Feedback	✗	f_c	f_c	f_c	f_c	f_c	f_c	f_c	f_c	f_c
Execution Feedback	✗	✗	f_e	f_e^*	✗	f_e	f_e^*	✗	f_e	f_e^*
Verbal Feedback	✗	✗	✗	✗	f_v	f_v	f_v	f_v^*	f_v^*	f_v^*
Closed-Source Models										
GPT-4-0613	46.0	46.0	60.3	**70.5**	46.0	**61.9**	**72.5**	**89.7**	**91.1**	**92.5**
GPT-4-Turbo-2024-04-09	48.0	48.0	56.7	63.8	48.0	58.6	68.1	84.7	87.5	88.5
GPT-4o-2024-05-13	**50.8**	**50.8**	**60.5**	67.6	**50.8**	60.8	69.6	82.3	84.9	86.2
Open-Source Models ($\geqslant$ 30B)										
DeepSeek-R1-Distill-Llama-70B (16K)	46.1	46.2	**61.7**	**72.7**	46.2	60.2	73.8	82.0	86.8	86.1
Llama-3.3-70B-Instruct	**47.6**	**47.7**	59.0	67.7	**47.7**	**61.5**	72.2	84.6	87.6	88.9
DeepSeek-R1-Distill-Qwen-32B (16K)	45.9	45.9	59.5	68.1	45.9	61.2	**74.0**	85.0	88.1	89.0
Qwen2.5-32B	45.8	45.9	50.4	53.9	46.0	54.8	62.6	84.7	85.5	87.5
Llama-3.1-70B-Instruct	45.4	45.4	56.2	64.8	45.4	59.5	70.8	**86.7**	**88.9**	**91.8**
DeepSeek-Coder-33B-Instruct	41.6	41.6	45.5	46.1	41.6	50.4	56.6	85.4	84.6	85.6
ReflectionCoder-DS-33B	41.6	41.6	45.3	44.9	41.6	51.4	57.2	81.4	81.8	84.2
Qwen1.5-72B-Chat	32.9	33.2	39.9	47.5	33.2	47.5	57.9	84.4	86.1	87.2
Qwen1.5-32B-Chat	32.0	32.0	41.1	45.3	32.0	44.6	54.3	75.9	61.8	57.1
CodeLlama-34B-Instruct	28.8	28.8	33.7	35.8	28.8	37.5	44.6	80.0	82.0	82.3
Open-Source Models ($<$ 30B)										
Llama-3.1-8B-Instruct	31.4	31.8	38.4	40.0	31.7	43.2	**51.8**	80.9	80.2	**83.7**
DeepSeek-Coder-V2-Lite-Instruct	38.3	38.3	**43.4**	**46.1**	38.3	**47.0**	51.4	76.3	75.8	76.9
DeepSeek-Coder-6.7B-Instruct	35.2	35.2	37.7	37.5	35.2	43.3	48.2	**82.8**	**82.5**	83.1
ReflectionCoder-DS-6.7B	37.4	37.4	39.6	40.7	37.4	44.7	50.4	79.1	79.6	78.9
CodeQwen1.5-7B-Chat	**39.3**	**39.6**	40.1	41.1	**39.5**	45.8	49.5	74.4	74.7	77.4
StarCoder2-15B-Instruct-v0.1	37.1	37.1	39.3	40.0	37.1	42.6	46.3	76.9	76.8	75.6
CodeLlama-13B-Instruct	28.4	28.4	29.7	30.0	28.4	35.1	41.1	69.0	70.7	71.6
CodeLlama-7B-Instruct	21.8	21.8	22.9	23.0	21.8	26.2	30.5	61.7	53.9	55.2

Training on a specific feedback combination can limit an LLM’s ability to utilize unseen combinations.

ConvCode
WORLD

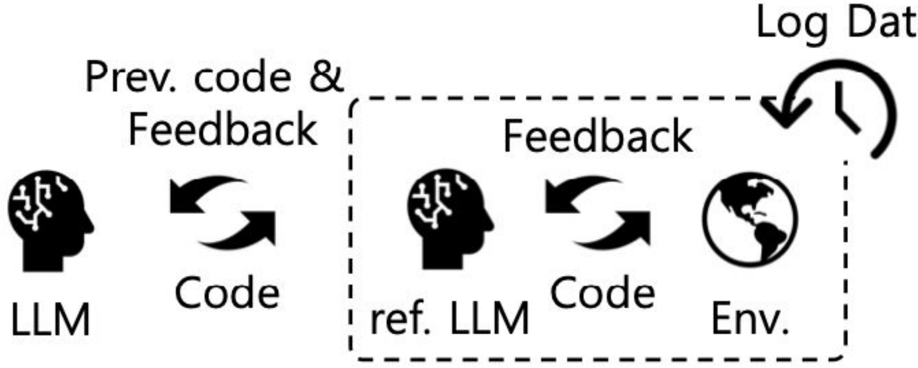
A Reproducible Environment with Diverse Feedback Combinations

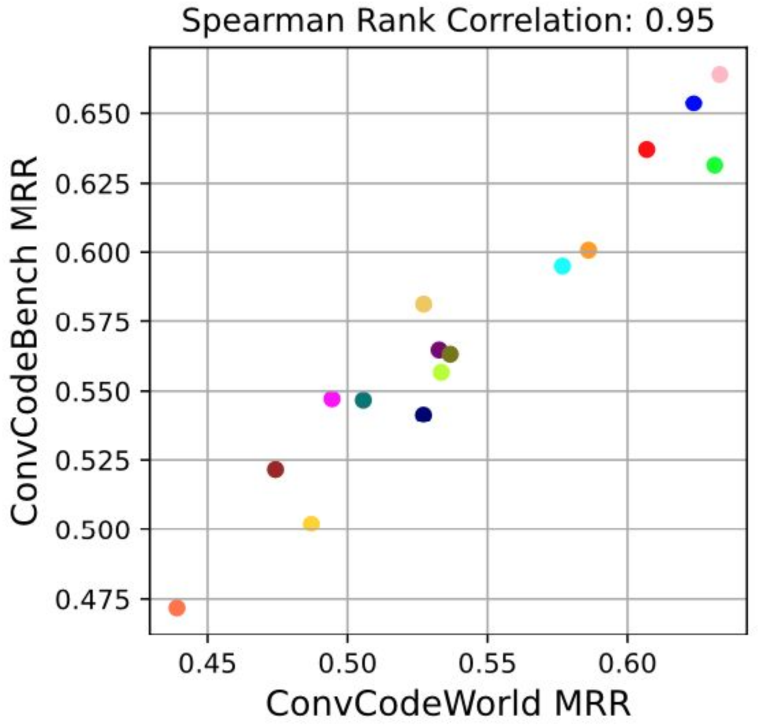


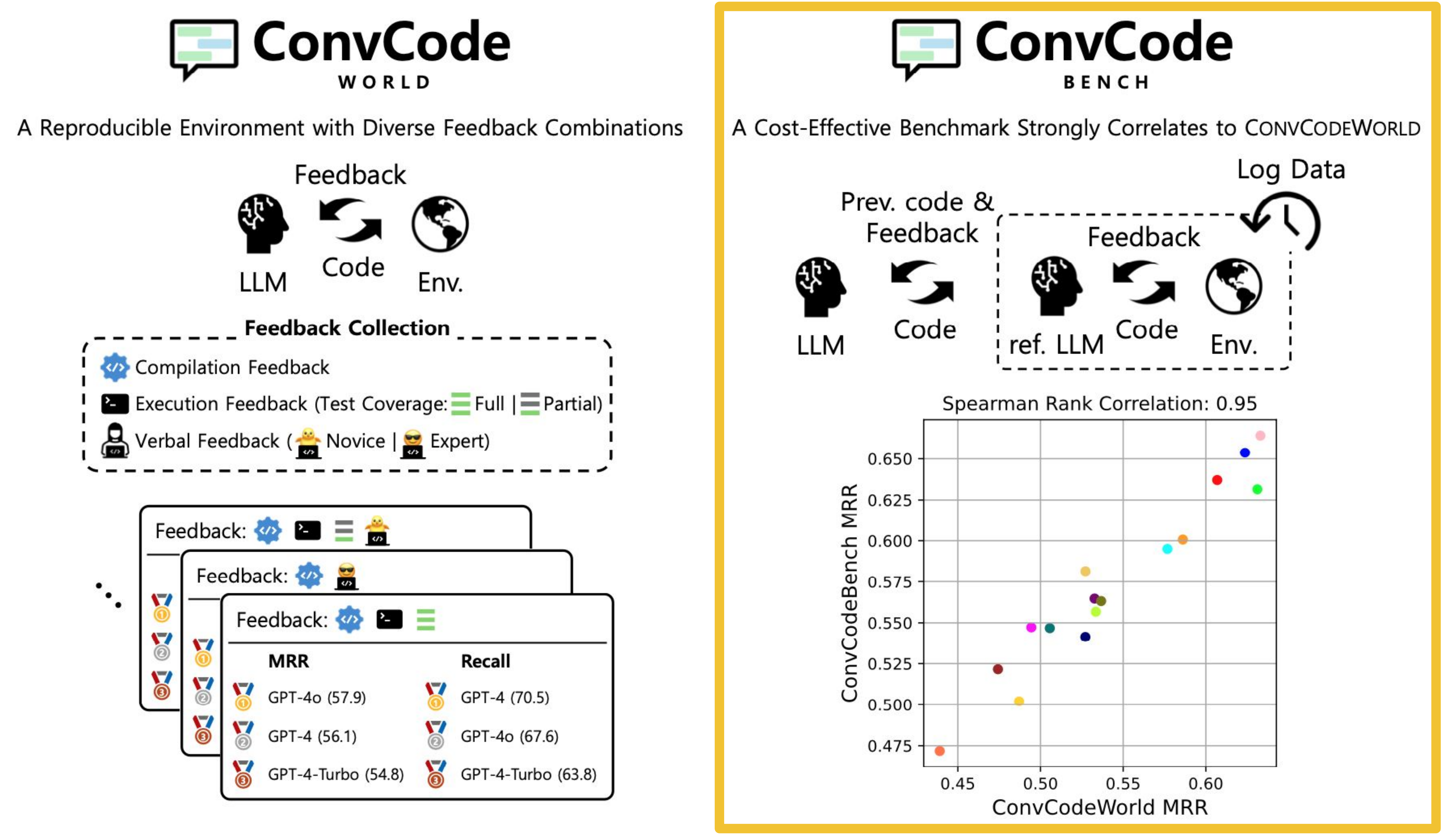
MRR	Recall
GPT-4o (57.9)	GPT-4 (70.5)
GPT-4 (56.1)	GPT-4o (67.6)
GPT-4-Turbo (54.8)	GPT-4-Turbo (63.8)

ConvCode
BENCH

A Cost-Effective Benchmark Strongly Correlates to ConvCodeWorld

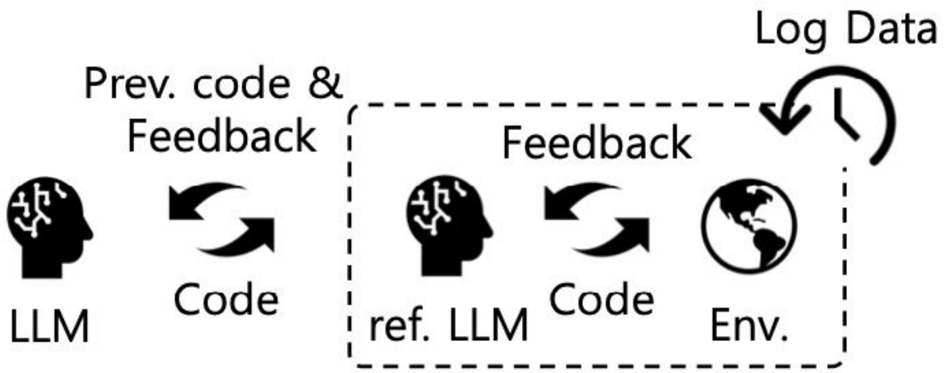






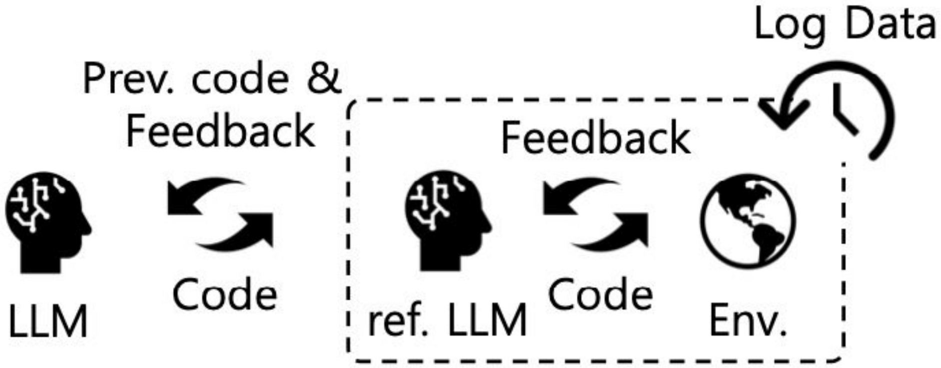
	MRR		Recall
①	GPT-4o (57.9)	①	GPT-4 (70.5)
②	GPT-4 (56.1)	②	GPT-4o (67.6)
③	GPT-4-Turbo (54.8)	③	GPT-4-Turbo (63.8)

ConvCodeBench



Pre-generated conversations by a fixed reference model

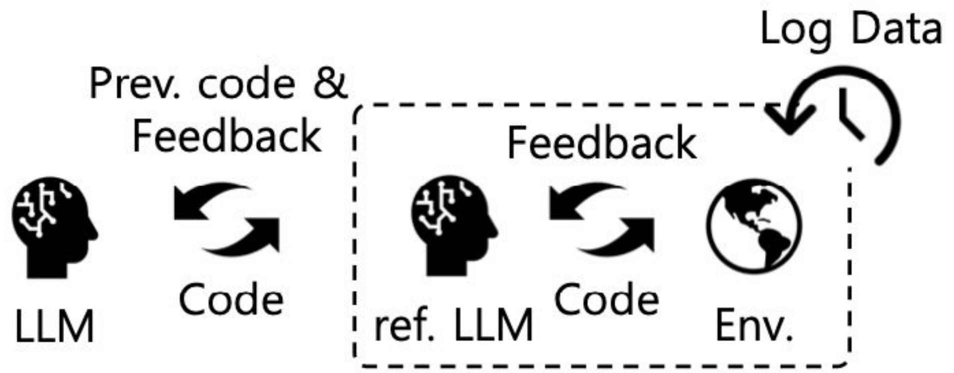
ConvCodeBench



Pre-generated conversations by a fixed reference model

- Elimination of Dependency on External LLMs or APIs for Verbal Feedback Generation

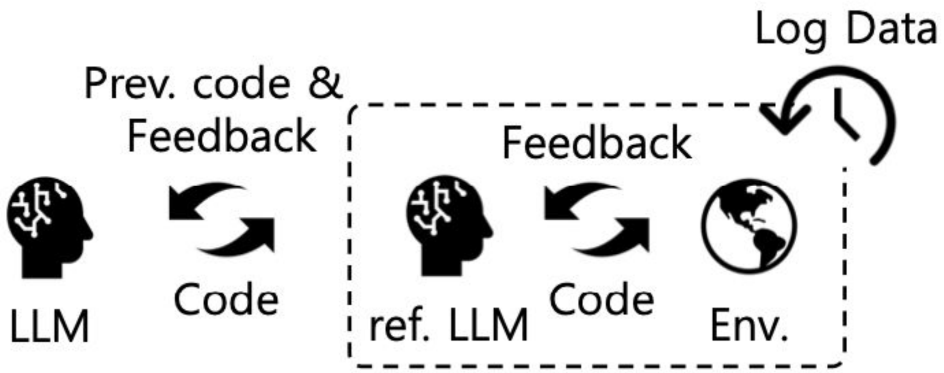
ConvCodeBench



Pre-generated conversations by a fixed reference model

- Elimination of Dependency on External LLMs or APIs for Verbal Feedback Generation
- Parallel Processing of Inference Calls

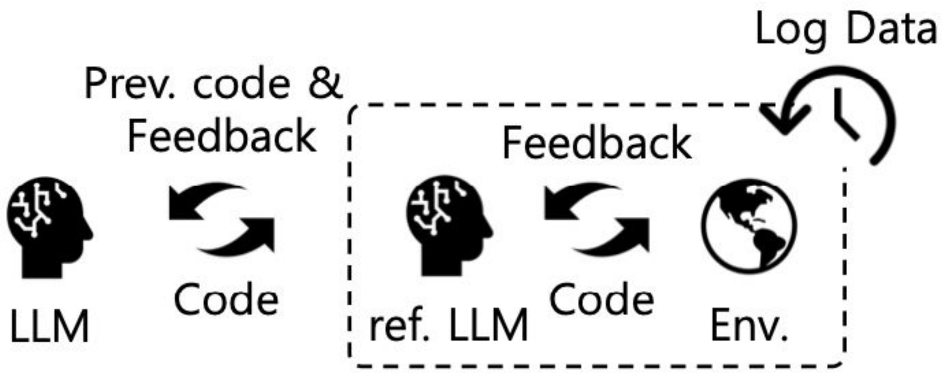
ConvCodeBench



Pre-generated conversations by a fixed reference model

- Elimination of Dependency on External LLMs or APIs for Verbal Feedback Generation
- Parallel Processing of Inference Calls
- Enhanced Reproducibility

ConvCodeBench

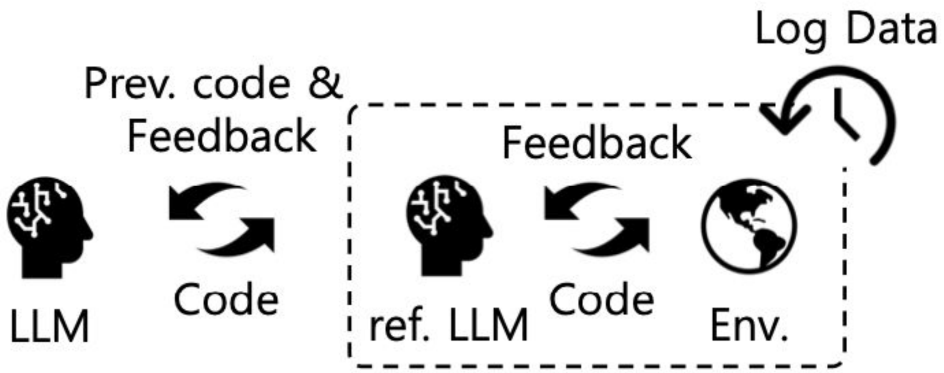


Pre-generated conversations by a fixed reference model

- Elimination of Dependency on External LLMs or APIs for Verbal Feedback Generation
- Parallel Processing of Inference Calls
- Enhanced Reproducibility

Question: *Can we ensure high correlation between ConvCodeWorld and ConvCodeBench?*

ConvCodeBench



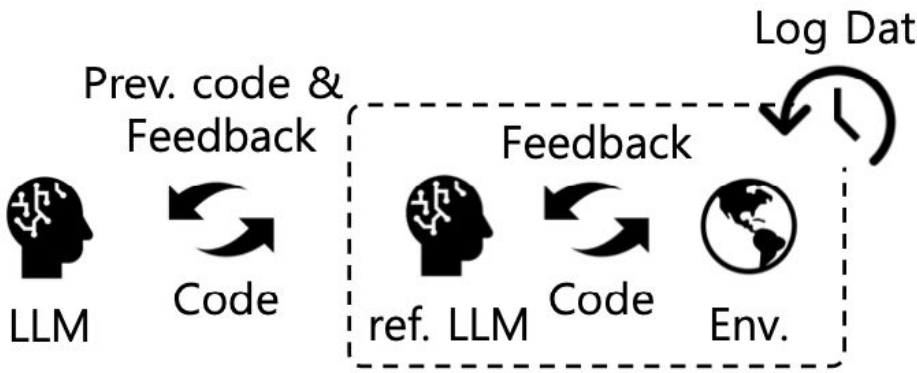
Pre-generated conversations by a fixed reference model

- Elimination of Dependency on External LLMs or APIs for Verbal Feedback Generation
- Parallel Processing of Inference Calls
- Enhanced Reproducibility

Question: *Can we ensure high correlation between ConvCodeWorld and ConvCodeBench?*

→ by selecting an appropriate reference model

ConvCodeBench

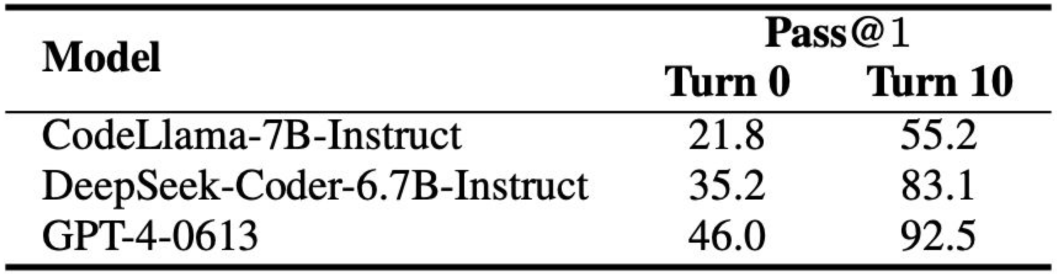


Pre-generated conversations by a fixed reference model

- Elimination of Dependency on External LLMs or APIs for Verbal Feedback Generation
- Parallel Processing of Inference Calls
- Enhanced Reproducibility

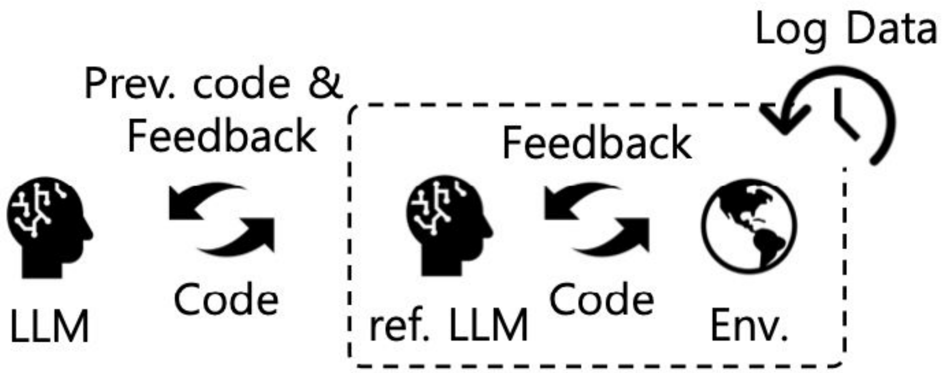
Question: *Can we ensure high correlation between ConvCodeWorld and ConvCodeBench?*

→ by selecting an appropriate reference model



Model	**Pass@1**	
	Turn 0	**Turn 10**
CodeLlama-7B-Instruct	21.8	55.2
DeepSeek-Coder-6.7B-Instruct	35.2	83.1
GPT-4-0613	46.0	92.5

ConvCodeBench

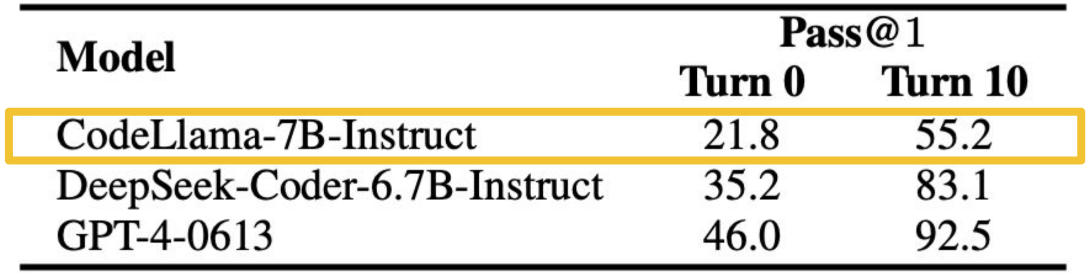


Pre-generated conversations by a fixed reference model

- Elimination of Dependency on External LLMs or APIs for Verbal Feedback Generation
- Parallel Processing of Inference Calls
- Enhanced Reproducibility

Question: *Can we ensure high correlation between ConvCodeWorld and ConvCodeBench?*

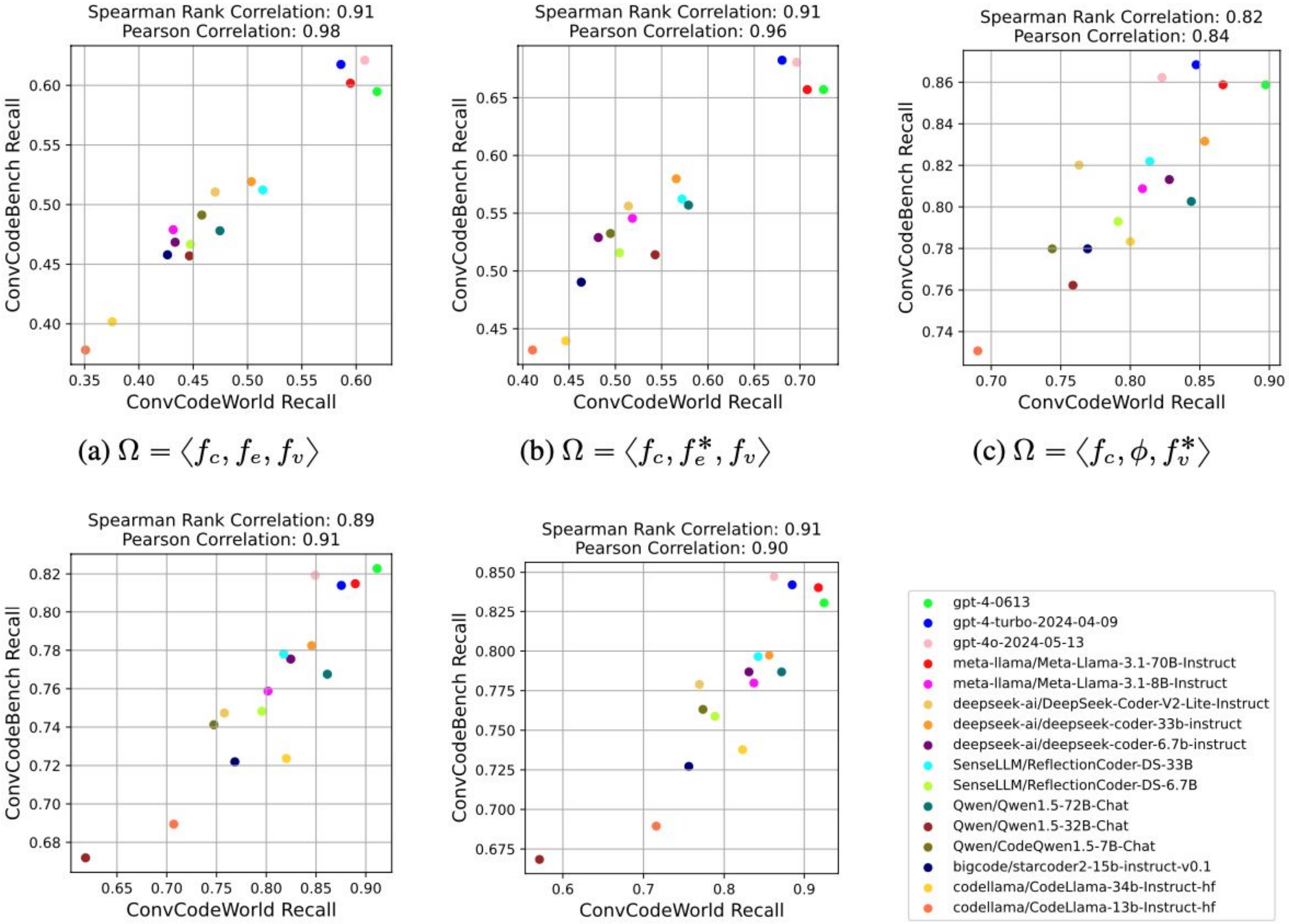
→ by selecting an appropriate reference model



Model	**Pass@1**	
	Turn 0	**Turn 10**
CodeLlama-7B-Instruct	21.8	55.2
DeepSeek-Coder-6.7B-Instruct	35.2	83.1
GPT-4-0613	46.0	92.5

Our hypothesis: *Use the weakest model*

ConvCodeBench



(a) $\Omega = \langle f_c, f_e, f_v \rangle$

(b) $\Omega = \langle f_c, f_e^*, f_v \rangle$

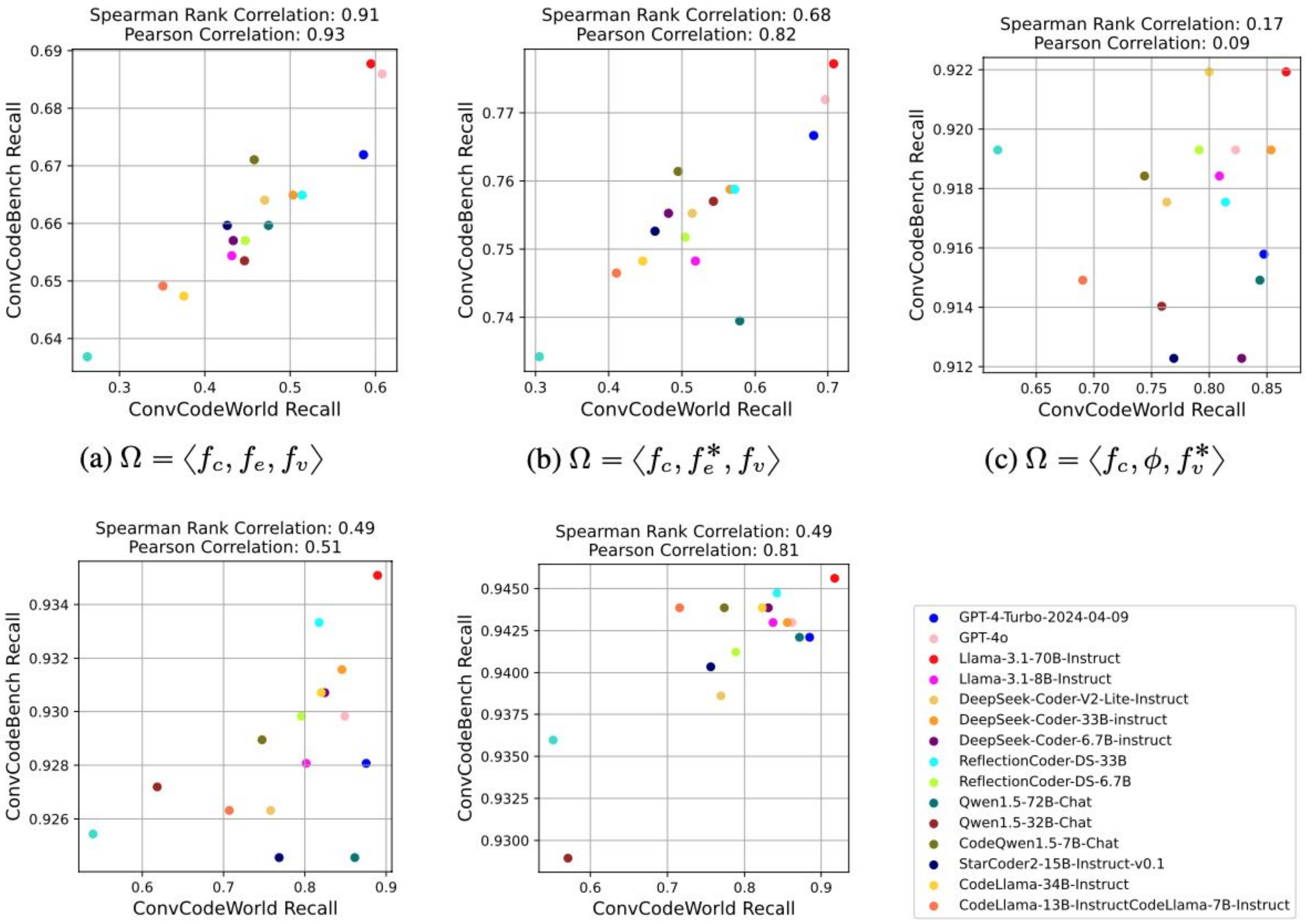
(c) $\Omega = \langle f_c, \phi, f_v^* \rangle$

(d) $\Omega = \langle f_c, f_e, f_v^* \rangle$

(e) $\Omega = \langle f_c, f_e^*, f_v^* \rangle$

Figure 6: Correlation between Recall on ConvCodeBench (ref. CodeLlama-7B-Instruct) and Recall on ConvCodeWorld with different feedback combinations Ω.

ConvCodeBench (ref. GPT-4-0613)



(a) $\Omega = \langle f_c, f_e, f_v \rangle$

(b) $\Omega = \langle f_c, f_e^*, f_v \rangle$

(c) $\Omega = \langle f_c, \phi, f_v^* \rangle$

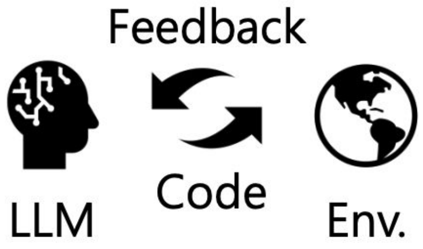
(d) $\Omega = \langle f_c, f_e, f_v^* \rangle$

(e) $\Omega = \langle f_c, f_e^*, f_v^* \rangle$

Figure 10: Correlation between Recall on CONVCODEBENCH (ref. GPT-4-0613) and Recall on CONVCODEWORLD with different feedback combinations Ω.

Summary

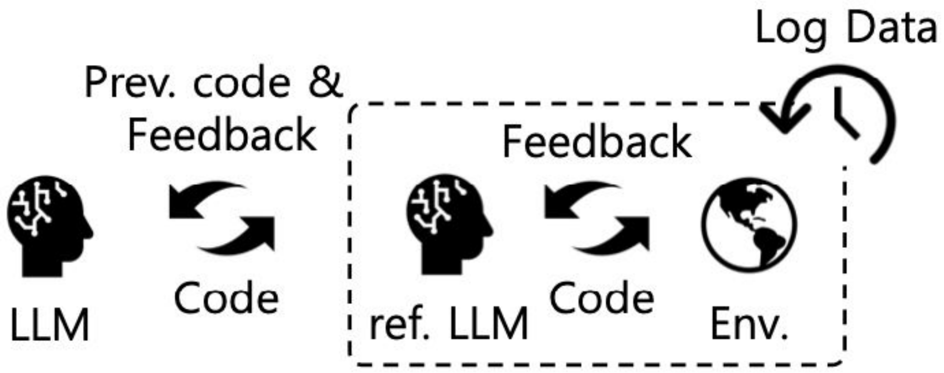
- **ConvCodeWorld**: a novel and reproducible environment for benchmarking interactive code generation
  - 9 distinct interactive code generation scenarios combining three types of feedback:

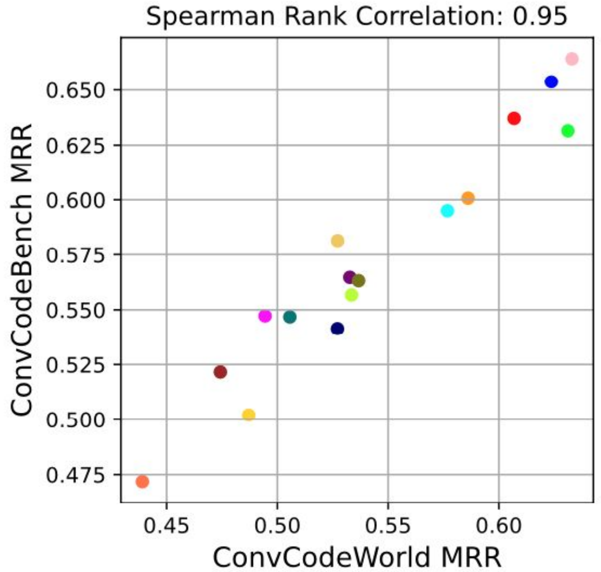


Feedback Collection

- Compilation Feedback
- Execution Feedback (Test Coverage: Full | Partial)
- Verbal Feedback (🐥 Novice | 😎 Expert)

- **ConvCodeBench**: a static version of benchmark that uses pre-generated feedback logs
  - eliminates the need for costly dynamic verbal feedback generation while maintaining strong Spearman's rank correlations (0.82 to 0.99) with ConvCodeWorld




Spearman Rank Correlation: 0.95

Any Questions?

Appendix

Experimental Results

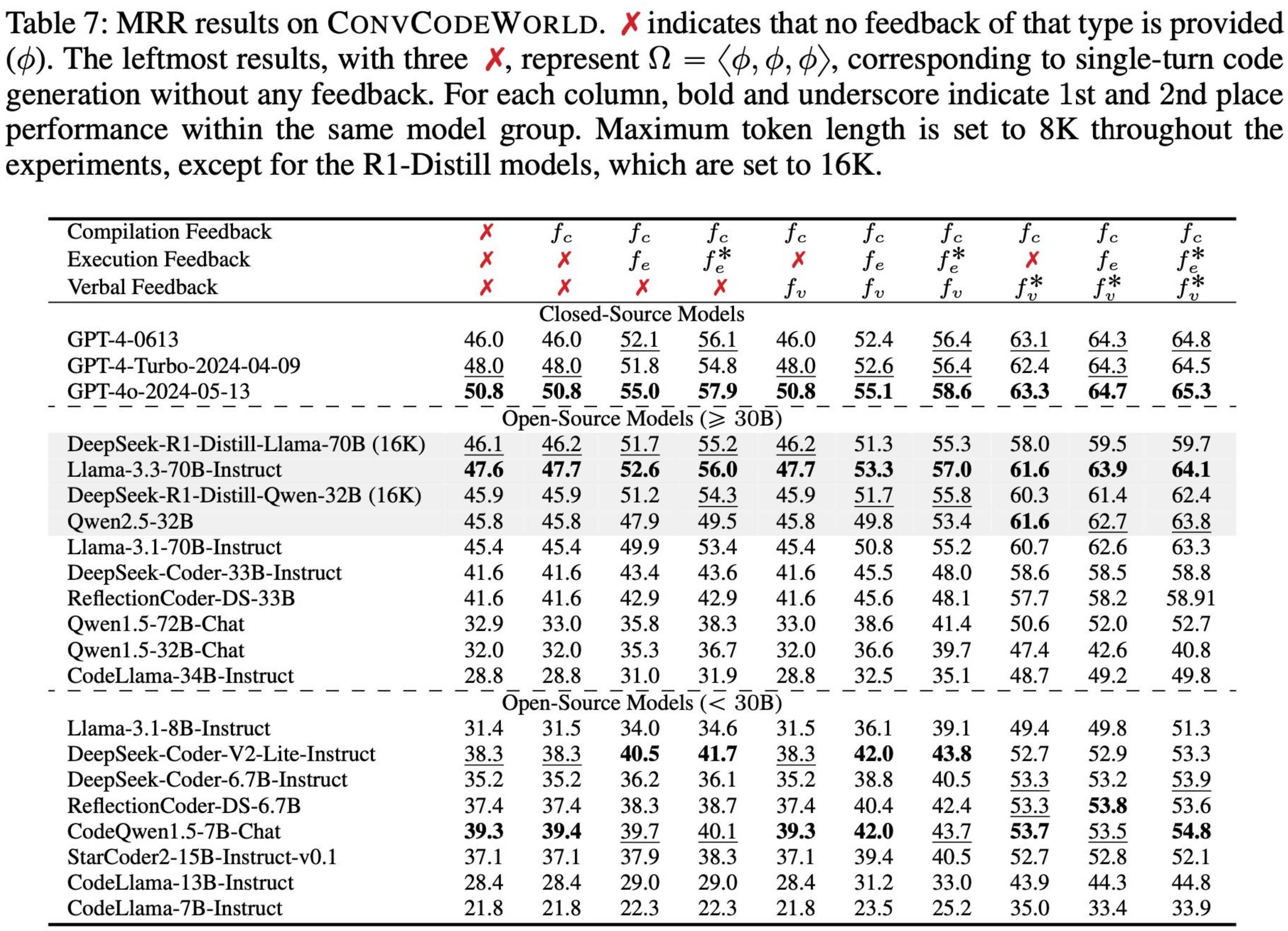
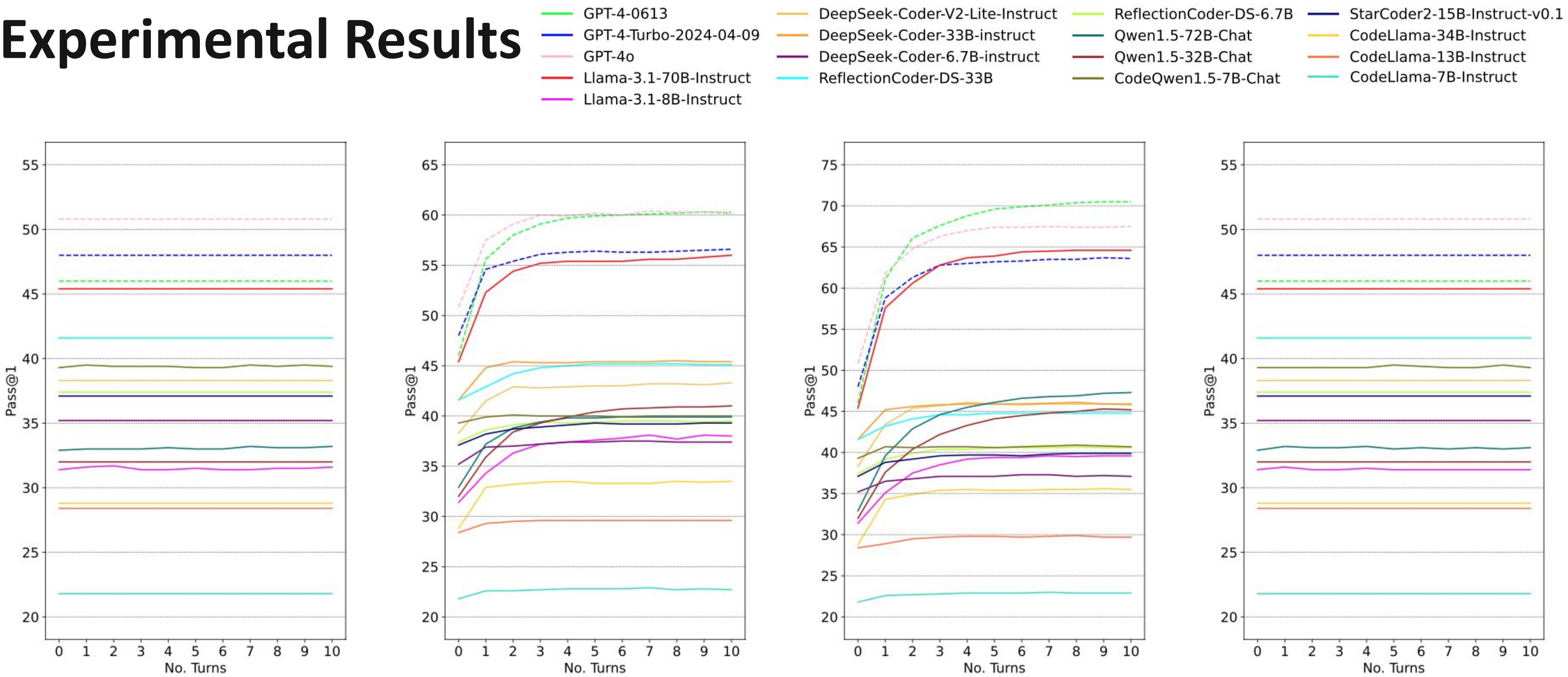


Table 7: MRR results on CONVCODEWORLD. ✗ indicates that no feedback of that type is provided (ϕ). The leftmost results, with three ✗, represent $\Omega = \langle \phi, \phi, \phi \rangle$, corresponding to single-turn code generation without any feedback. For each column, bold and underscore indicate 1st and 2nd place performance within the same model group. Maximum token length is set to 8K throughout the experiments, except for the R1-Distill models, which are set to 16K.

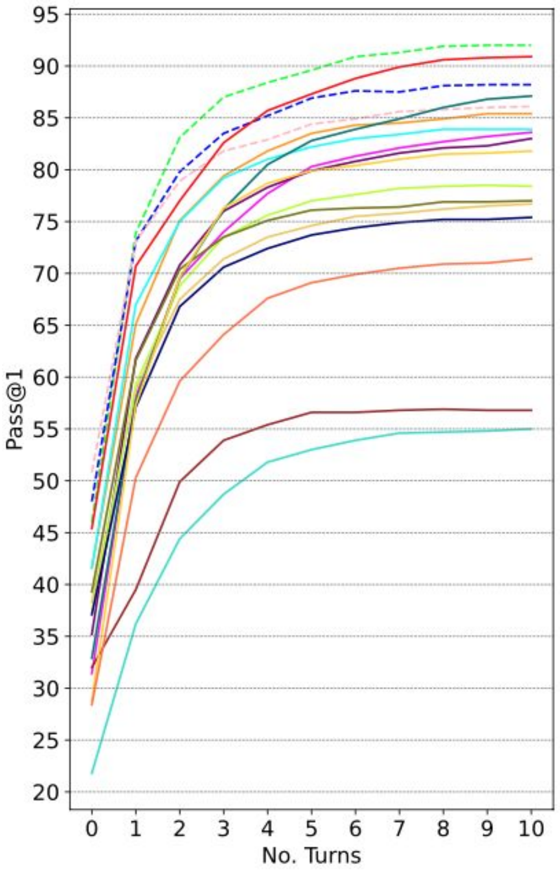
Compilation Feedback	✗	f_c	f_c	f_c	f_c	f_c	f_c	f_c	f_c	f_c
Execution Feedback	✗	✗	f_e	f_e^*	✗	f_e	f_e^*	✗	f_e	f_e^*
Verbal Feedback	✗	✗	✗	✗	f_v	f_v	f_v	f_v^*	f_v^*	f_v^*
Closed-Source Models										
GPT-4-0613	46.0	46.0	52.1	56.1	46.0	52.4	56.4	63.1	64.3	64.8
GPT-4-Turbo-2024-04-09	48.0	48.0	51.8	54.8	48.0	52.6	56.4	62.4	64.3	64.5
GPT-4o-2024-05-13	**50.8**	**50.8**	**55.0**	**57.9**	**50.8**	**55.1**	**58.6**	**63.3**	**64.7**	**65.3**
Open-Source Models ($\geqslant$ 30B)										
DeepSeek-R1-Distill-Llama-70B (16K)	46.1	46.2	51.7	55.2	46.2	51.3	55.3	58.0	59.5	59.7
Llama-3.3-70B-Instruct	**47.6**	**47.7**	**52.6**	**56.0**	**47.7**	**53.3**	**57.0**	**61.6**	**63.9**	**64.1**
DeepSeek-R1-Distill-Qwen-32B (16K)	45.9	45.9	51.2	54.3	45.9	51.7	55.8	60.3	61.4	62.4
Qwen2.5-32B	45.8	45.8	47.9	49.5	45.8	49.8	53.4	**61.6**	62.7	63.8
Llama-3.1-70B-Instruct	45.4	45.4	49.9	53.4	45.4	50.8	55.2	60.7	62.6	63.3
DeepSeek-Coder-33B-Instruct	41.6	41.6	43.4	43.6	41.6	45.5	48.0	58.6	58.5	58.8
ReflectionCoder-DS-33B	41.6	41.6	42.9	42.9	41.6	45.6	48.1	57.7	58.2	58.91
Qwen1.5-72B-Chat	32.9	33.0	35.8	38.3	33.0	38.6	41.4	50.6	52.0	52.7
Qwen1.5-32B-Chat	32.0	32.0	35.3	36.7	32.0	36.6	39.7	47.4	42.6	40.8
CodeLlama-34B-Instruct	28.8	28.8	31.0	31.9	28.8	32.5	35.1	48.7	49.2	49.8
Open-Source Models ($<$ 30B)										
Llama-3.1-8B-Instruct	31.4	31.5	34.0	34.6	31.5	36.1	39.1	49.4	49.8	51.3
DeepSeek-Coder-V2-Lite-Instruct	38.3	38.3	**40.5**	**41.7**	38.3	**42.0**	**43.8**	52.7	52.9	53.3
DeepSeek-Coder-6.7B-Instruct	35.2	35.2	36.2	36.1	35.2	38.8	40.5	53.3	53.2	53.9
ReflectionCoder-DS-6.7B	37.4	37.4	38.3	38.7	37.4	40.4	42.4	53.3	**53.8**	53.6
CodeQwen1.5-7B-Chat	**39.3**	**39.4**	39.7	40.1	**39.3**	**42.0**	43.7	**53.7**	53.5	**54.8**
StarCoder2-15B-Instruct-v0.1	37.1	37.1	37.9	38.3	37.1	39.4	40.5	52.7	52.8	52.1
CodeLlama-13B-Instruct	28.4	28.4	29.0	29.0	28.4	31.2	33.0	43.9	44.3	44.8
CodeLlama-7B-Instruct	21.8	21.8	22.3	22.3	21.8	23.5	25.2	35.0	33.4	33.9

Experimental Results



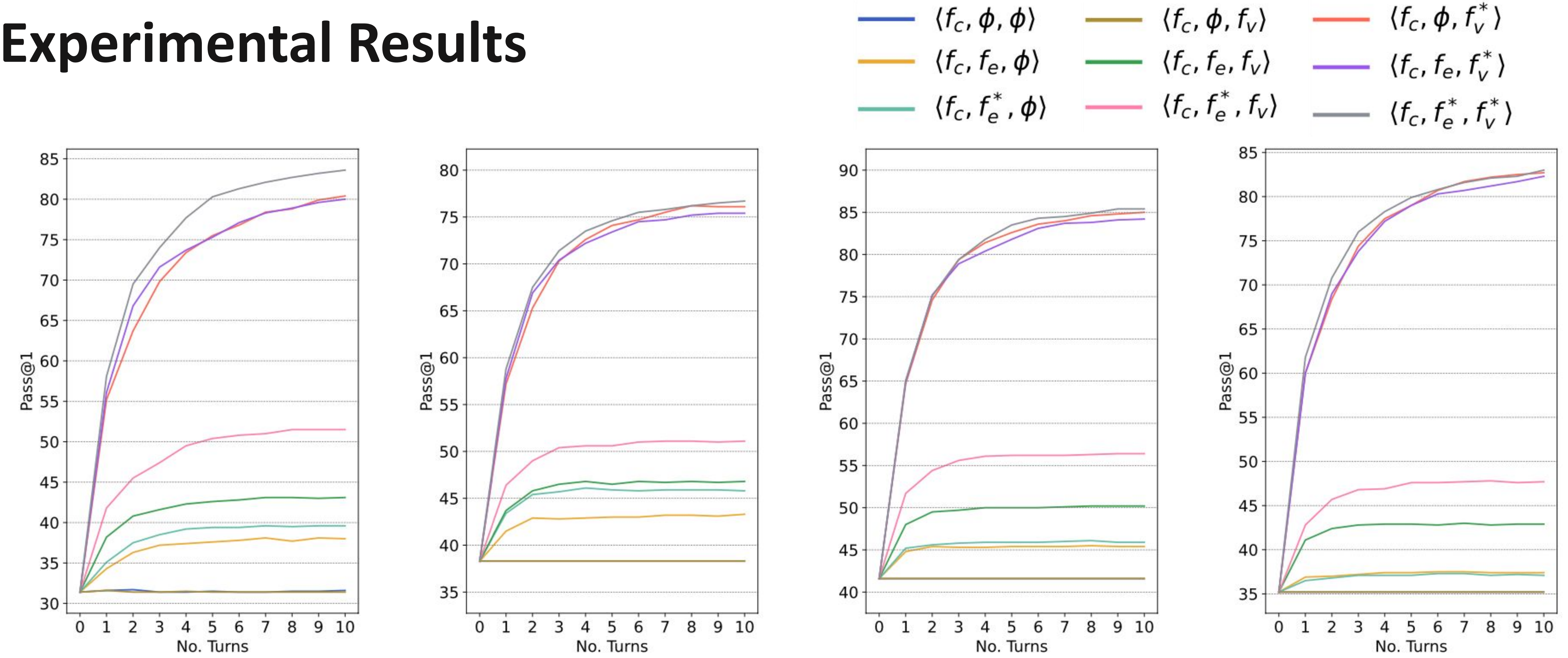
(a) $\Omega = \langle f_c, \phi, \phi \rangle$ (b) $\Omega = \langle f_c, f_e, \phi \rangle$ (c) $\Omega = \langle f_c, f_e^*, \phi \rangle$ (d) $\Omega = \langle f_c, \phi, f_v \rangle$

Experimental Results



(i) $\Omega = \langle f_c, f_e^*, f_v^* \rangle$

Experimental Results



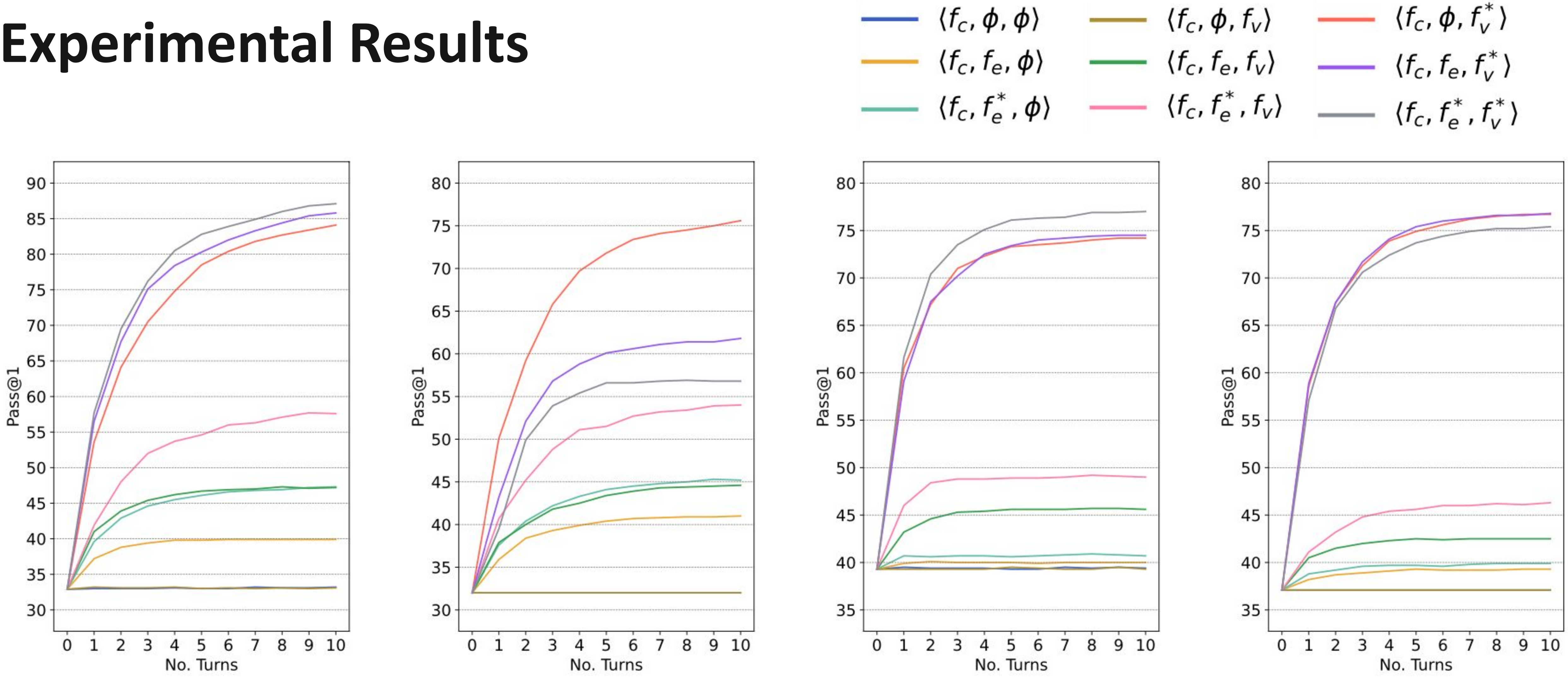
(e) Llama-3.1-8B-Instruct

(f) DeepSeek-Coder-V2-Lite-Instruct

(g) DeepSeek-Coder-33B-Instruct

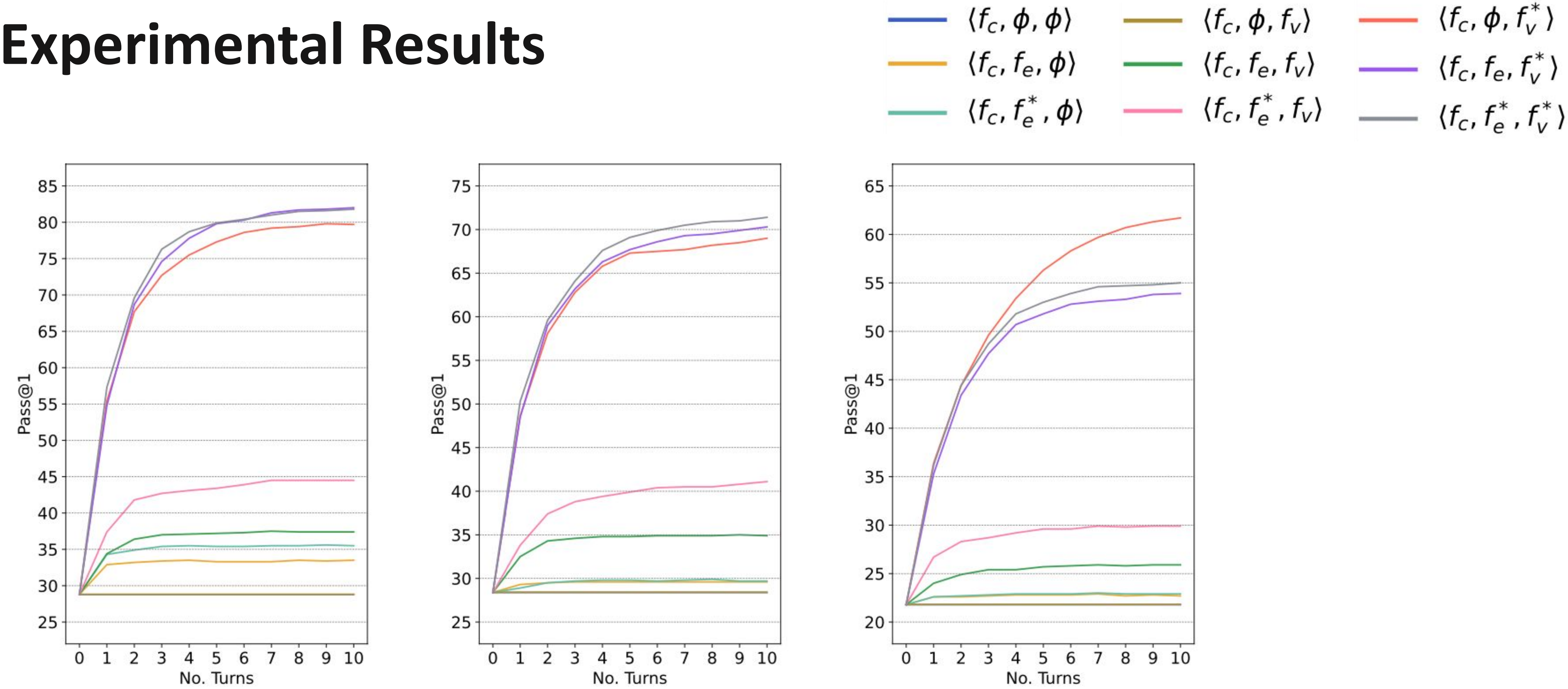
(h) DeepSeek-Coder-6.7B-Instruct

Experimental Results



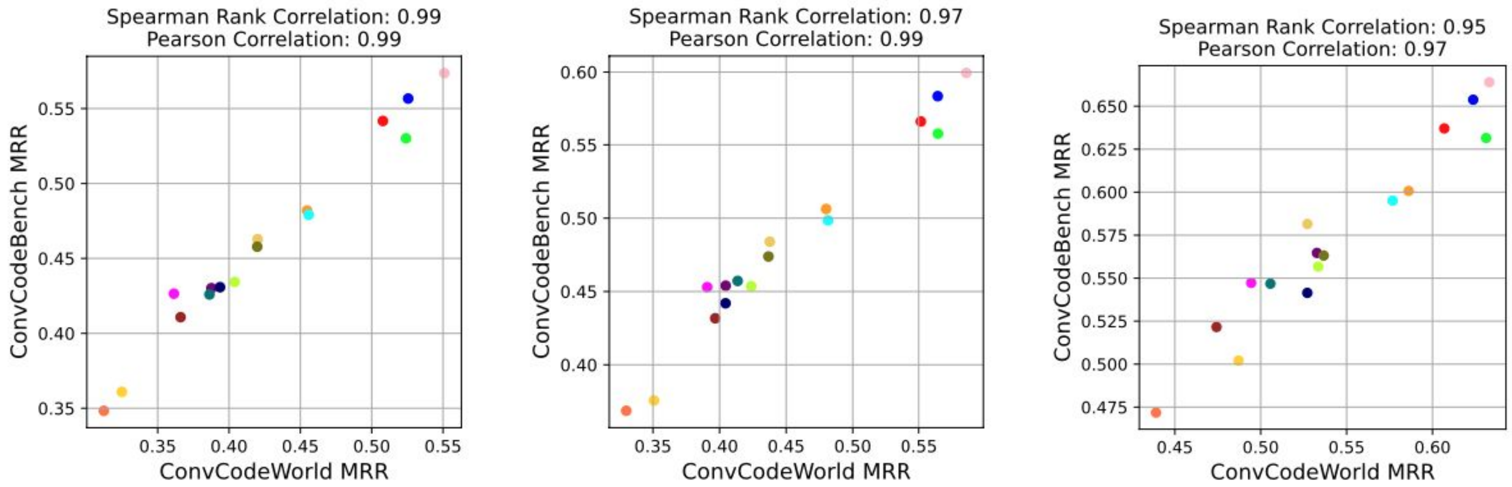
(a) Qwen1.5-72B-Chat (b) Qwen1.5-32B-Chat (c) CodeQwen1.5-7B-Chat (d) StarCoder2-15B-Instruct-v0.1

Experimental Results



(e) CodeLlama-34B-Instruct (f) CodeLlama-13B-Instruct (g) CodeLlama-7B-Instruct

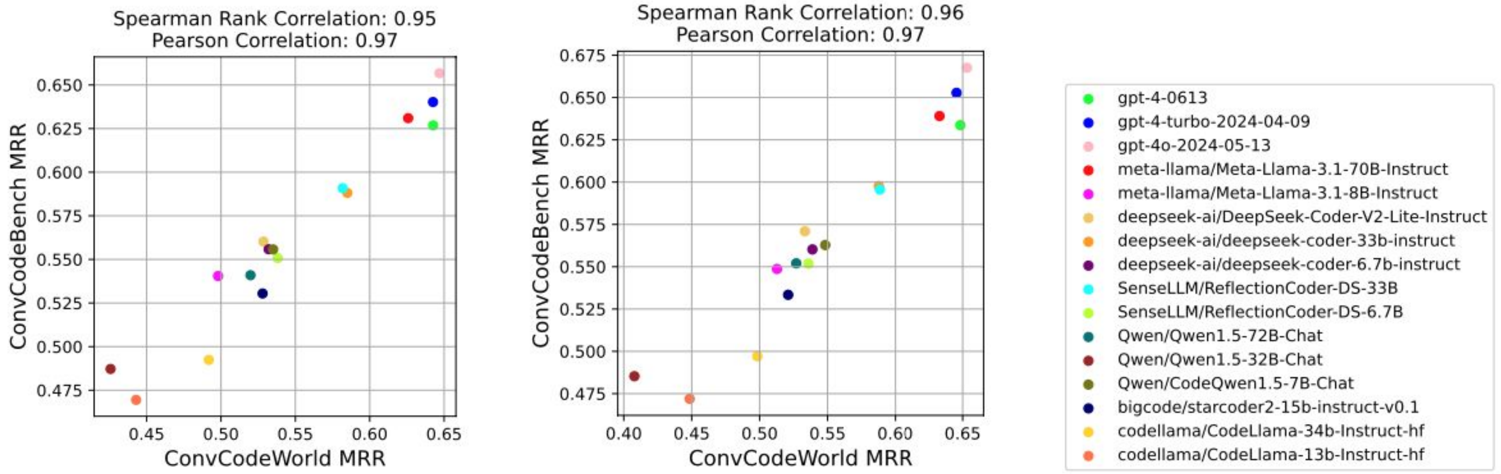
ConvCodeBench



(a) $\Omega = \langle f_c, f_e, f_v \rangle$

(b) $\Omega = \langle f_c, f_e^*, f_v \rangle$

(c) $\Omega = \langle f_c, \phi, f_v^* \rangle$

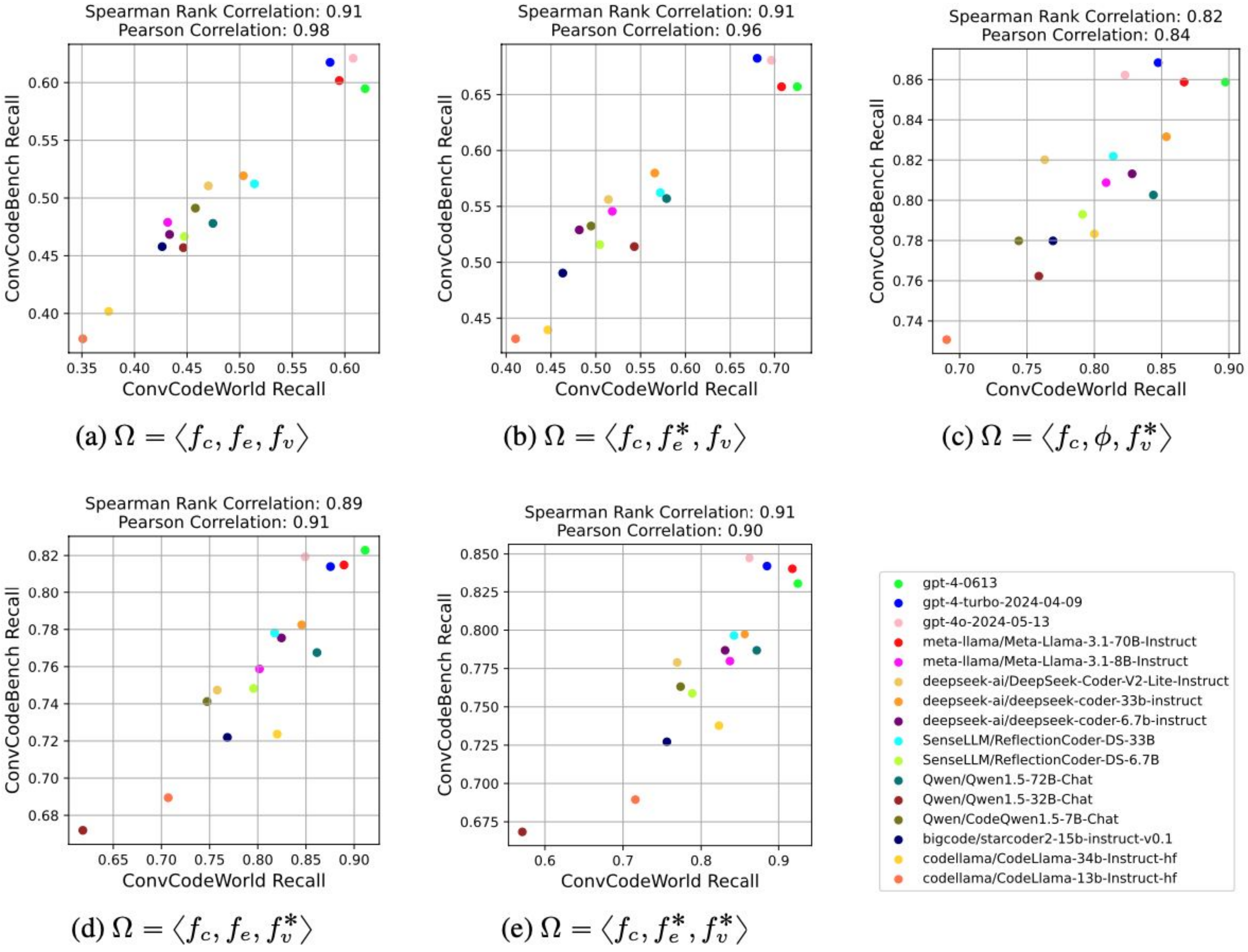


(d) $\Omega = \langle f_c, f_e, f_v^* \rangle$

(e) $\Omega = \langle f_c, f_e^*, f_v^* \rangle$

Figure 2: Correlation between MRR on CONVCODEBENCH (ref. CodeLlama-7B-Instruct) and MRR on CONVCODEWORLD with different feedback combinations Ω.

ConvCodeBench



(a) $\Omega = \langle f_c, f_e, f_v \rangle$

(b) $\Omega = \langle f_c, f_e^*, f_v \rangle$

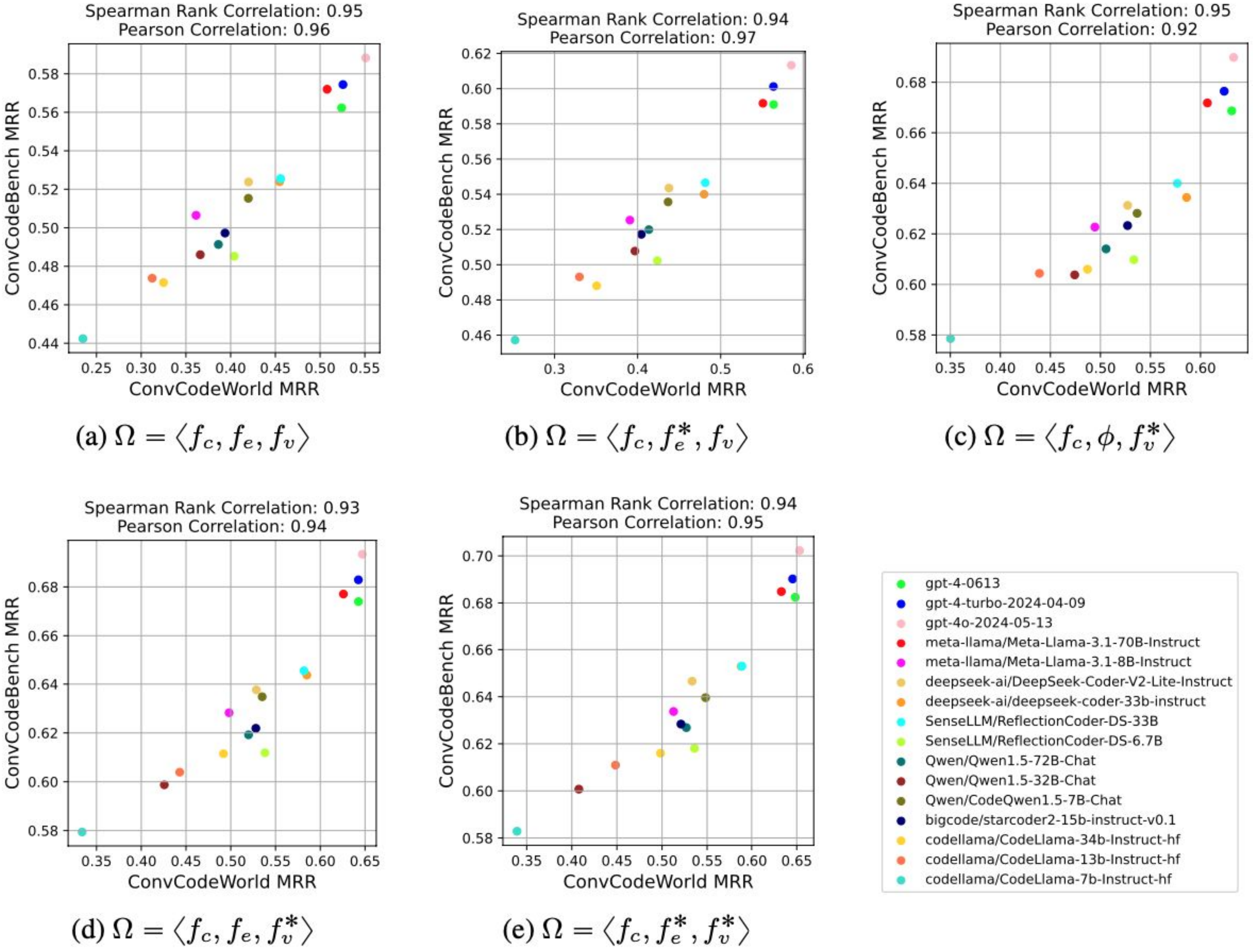
(c) $\Omega = \langle f_c, \phi, f_v^* \rangle$

(d) $\Omega = \langle f_c, f_e, f_v^* \rangle$

(e) $\Omega = \langle f_c, f_e^*, f_v^* \rangle$

Figure 6: Correlation between Recall on CONVCODEBENCH (ref. CodeLlama-7B-Instruct) and Recall on CONVCODEWORLD with different feedback combinations Ω.

ConvCodeBench



Spearman Rank Correlation: 0.95
Pearson Correlation: 0.96

(a) $\Omega = \langle f_c, f_e, f_v \rangle$

Spearman Rank Correlation: 0.94
Pearson Correlation: 0.97

(b) $\Omega = \langle f_c, f_e^*, f_v \rangle$

Spearman Rank Correlation: 0.95
Pearson Correlation: 0.92

(c) $\Omega = \langle f_c, \phi, f_v^* \rangle$

Spearman Rank Correlation: 0.93
Pearson Correlation: 0.94

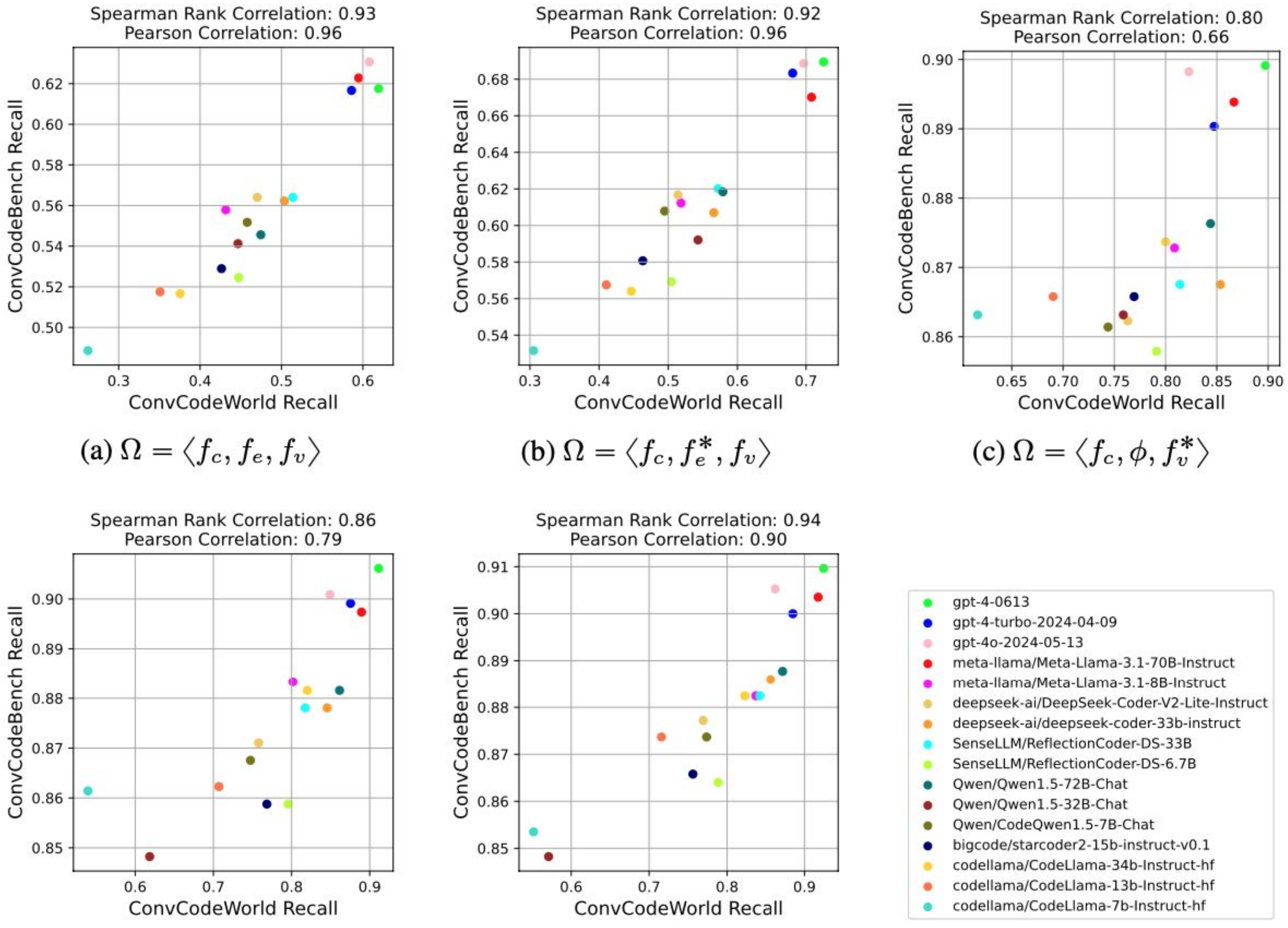
(d) $\Omega = \langle f_c, f_e, f_v^* \rangle$

Spearman Rank Correlation: 0.94
Pearson Correlation: 0.95

(e) $\Omega = \langle f_c, f_e^*, f_v^* \rangle$

Figure 7: Correlation between MRR on ConvCodeBench (ref. DeepSeek-Coder-6.7B-Instruct) and MRR on ConvCodeWorld with different feedback combinations Ω.

ConvCodeBench



(a) $\Omega = \langle f_c, f_e, f_v \rangle$

(b) $\Omega = \langle f_c, f_e^*, f_v \rangle$

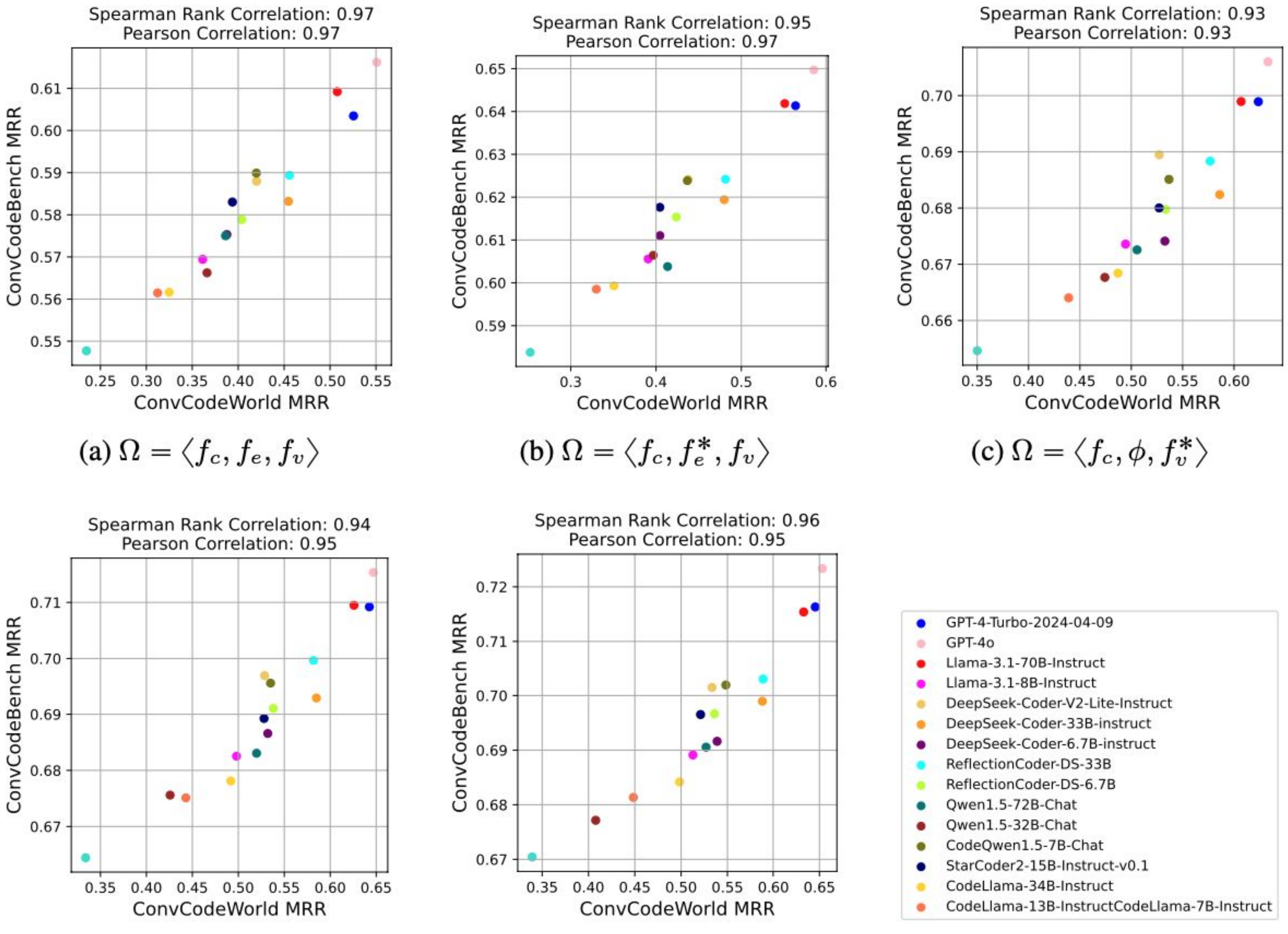
(c) $\Omega = \langle f_c, \phi, f_v^* \rangle$

(d) $\Omega = \langle f_c, f_e, f_v^* \rangle$

(e) $\Omega = \langle f_c, f_e^*, f_v^* \rangle$

Figure 8: Correlation between Recall on ConvCodeBench (ref. DeepSeek-Coder-6.7B-Instruct) and Recall on ConvCodeWorld with different feedback combinations Ω.

ConvCodeBench

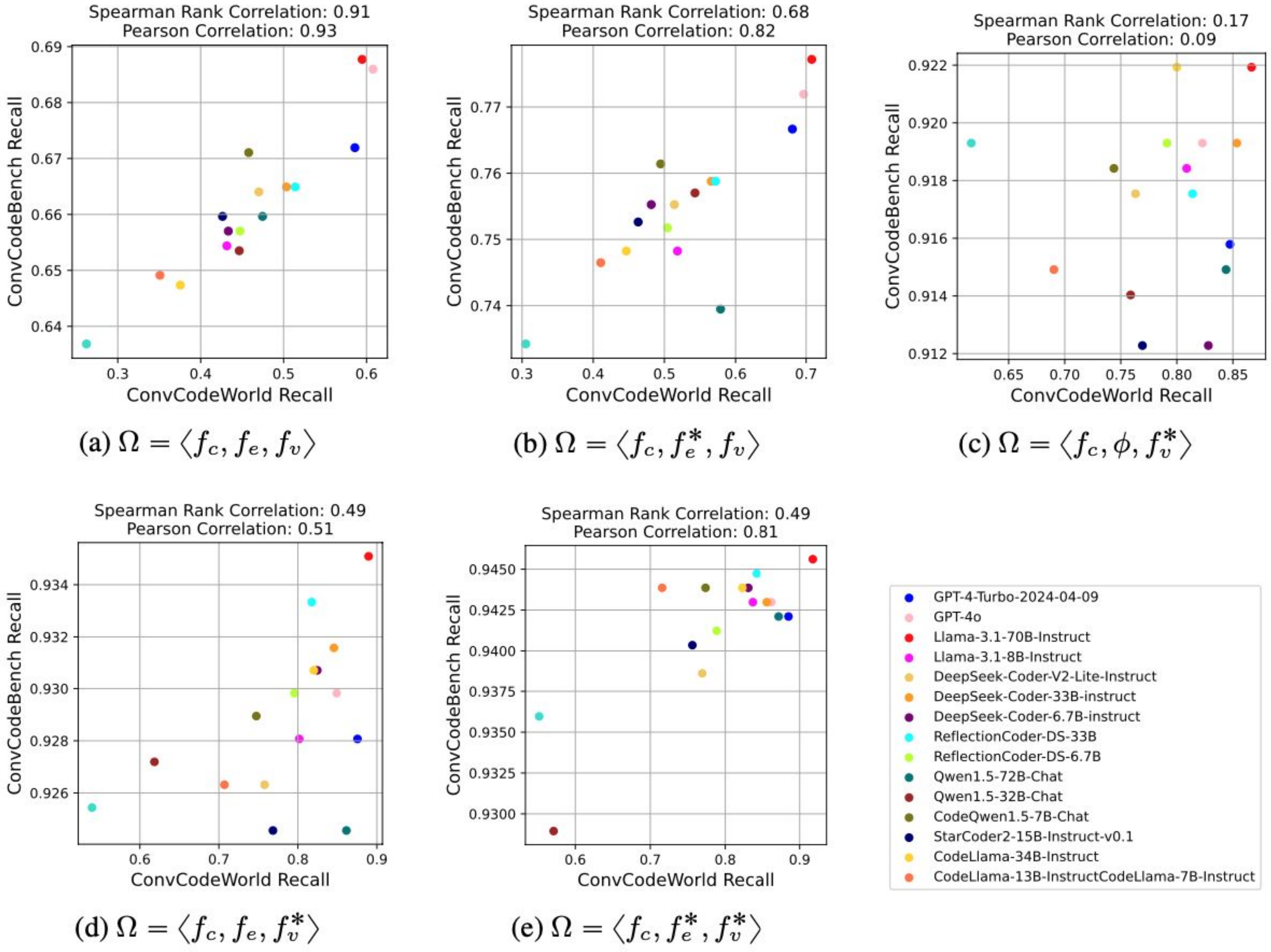


(a) $\Omega = \langle f_c, f_e, f_v \rangle$ (b) $\Omega = \langle f_c, f_e^*, f_v \rangle$ (c) $\Omega = \langle f_c, \phi, f_v^* \rangle$

(d) $\Omega = \langle f_c, f_e, f_v^* \rangle$ (e) $\Omega = \langle f_c, f_e^*, f_v^* \rangle$

Figure 9: Correlation between MRR on CONVCODEBENCH (ref. GPT-4-0613) and MRR on CONVCODEWORLD with different feedback combinations Ω.

ConvCodeBench



Spearman Rank Correlation: 0.91
Pearson Correlation: 0.93

(a) $\Omega = \langle f_c, f_e, f_v \rangle$

Spearman Rank Correlation: 0.68
Pearson Correlation: 0.82

(b) $\Omega = \langle f_c, f_e^*, f_v \rangle$

Spearman Rank Correlation: 0.17
Pearson Correlation: 0.09

(c) $\Omega = \langle f_c, \phi, f_v^* \rangle$

Spearman Rank Correlation: 0.49
Pearson Correlation: 0.51

(d) $\Omega = \langle f_c, f_e, f_v^* \rangle$

Spearman Rank Correlation: 0.49
Pearson Correlation: 0.81

(e) $\Omega = \langle f_c, f_e^*, f_v^* \rangle$

Figure 10: Correlation between Recall on CONVCODEBENCH (ref. GPT-4-0613) and Recall on CONVCODEWORLD with different feedback combinations Ω.

Table 14: Pass@1 results over different model combinations of expert-level verbal feedback f_v^* generation and code generation on CONVCODEWORLD where $\Omega = \langle f_c, \phi, f_v^* \rangle$ and the total number of turns $n = 1$. For each column, bold and underscore indicate 1st and 2nd place performance while keeping the code generation model fixed.

f_v^* Generation	Code Generation	
	GPT-4o-2024-05-13	Llama-3.1-70B-Instruct
w/o Feedback	50.8	45.4
GPT-4o-2024-05-13	<u>64.2</u>	**65.1**
Llama-3.1-70B-Instruct	**65.8**	<u>62.1</u>